Attorney Docket No. 826.1880

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:

Takahisa HATAKEYAMA et al.

Application No.: (Unassigned) Group Art Unit:

Filed: July 9, 2003 Examiner:

For: OPEN GENERIC TAMPER RESISTANT CPU AND APPLICATION SYSTEM THEREOF

SUBMISSION OF CERTIFIED COPY OF PRIOR FOREIGN APPLICATION IN ACCORDANCE WITH THE REQUIREMENTS OF 37 C.F.R. § 1.55

Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Sir:

In accordance with the provisions of 37 C.F.R. § 1.55, the applicant(s) submit(s) herewith a certified copy of the following foreign application:

Japanese Patent Application No(s). PCT/JP02/06955

Filed: July 9, 2002

It is respectfully requested that the applicant(s) be given the benefit of the foreign filing date(s) as evidenced by the certified papers attached hereto, in accordance with the requirements of 35 U.S.C. § 119.

Respectfully submitted,

STAAS & HALSEY LLP

Date: 7/9/3 By: _____

J. Randall Beckers
Registration No. 30,358

1201 New York Ave, N.W., Suite 700
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501

JAPAN PATENT OFFICE

This is to certify that the annexed is a true copy of the following application as filed with this office.

Date of Application: July 9, 2002

Application Number: PCT/JP02/06955

Applicant(s): FUJITSU LIMITED

April 8, 2003

Commissioner,
Japan Patent Office Shinichiro Ota

Certificate No. H14-500078

日 本 国 特 許 庁

JAPAN PATENT OFFICE

別紙添付の書類は下記の出願書類の謄本に相違ないことを証明する。
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００２年 ７月 ９日

出願番号
Application Number: ＰＣＴ／ＪＰ０２／０６９５５

出願人
Applicant (s): 富士通株式会社

２００３年 ４月 ８日

特許庁長官
Commissioner,
Japan Patent Office 太田信一郎

出証平 １４－５００７８

受理官庁用写し

特許協力条約に基づく国際出願願書 0251475/2555
原本（出願用）- 印刷日時 2002年07月09日 (09.07.2002) 火曜日 14時42分35秒

0	受理官庁記入欄	PCT/JP02/06955
0-1	国際出願番号.	
0-2	国際出願日	09.07.02
0-3	(受付印)	PCT International Application 日本国特許庁
0-4	様式-PCT/RO/101 この特許協力条約に基づく国際出願願書は、	
0-4-1	右記によって作成された。	PCT-EASY Version 2.92 (updated 01.06.2002)
0-5	申立て 出願人は、この国際出願が特許協力条約に従って処理されることを請求する。	
0-6	出願人によって指定された受理官庁	日本国特許庁 (RO/JP)
0-7	出願人又は代理人の書類記号	0251475/2555
I	発明の名称	開放型汎用耐攻撃ＣＰＵ及びその応用システム
II	出願人	
II-1	この欄に記載した者は	出願人である (applicant only)
II-2	右の指定国についての出願人である。	すべての指定国 (all designated States)
II-4ja	名称	富士通株式会社
II-4en	Name	FUJITSU LIMITED
II-5ja	あて名:	211-8588 日本国 神奈川県 川崎市 中原区上小田中４丁目１番１号
II-5en	Address:	1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8588 Japan
II-6	国籍（国名）	日本国 JP
II-7	住所（国名）	日本国 JP
II-8	電話番号	044-754-3798
II-9	ファクシミリ番号	044-754-3536
III-1	その他の出願人又は発明者	
III-1-1	この欄に記載した者は	発明者である (inventor only)
III-1-4ja	氏名(姓名)	畠山 卓久
III-1-4en	Name (LAST, First)	HATAKEYAMA, Takahisa
III-1-5ja	あて名:	211-8588 日本国 神奈川県 川崎市 中原区上小田中4丁目1番１号 富士通株式会社内
III-1-5en	Address:	c/o FUJITSU LIMITED, 1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8588 Japan

特許協力条約に基づく国際出願願書 0251475/2555

III-2	その他の出願人又は発明者	
III-2-1	この欄に記載した者は	発明者である (inventor only)
III-2-4ja	氏名(姓名)	丸山 秀史
III-2-4en	Name (LAST, First)	MURUYAMA, Hidefumi
III-2-5ja	あて名:	211-8588 日本国 神奈川県 川崎市 中原区上小田中4丁目1番１号 富士通株式会社内
III-2-5en	Address:	c/o FUJITSU LIMITED, 1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8588 Japan
III-3	その他の出願人又は発明者	
III-3-1	この欄に記載した者は	発明者である (inventor only)
III-3-4ja	氏名(姓名)	牛若 恵一
III-3-4en	Name (LAST, First)	USHIWAKA, Keiichi
III-3-5ja	あて名:	211-8588 日本国 神奈川県 川崎市 中原区上小田中4丁目1番１号 富士通株式会社内
III-3-5en	Address:	c/o FUJITSU LIMITED, 1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8588 Japan
III-4	その他の出願人又は発明者	
III-4-1	この欄に記載した者は	発明者である (inventor only)
III-4-4ja	氏名(姓名)	長谷部 高行
III-4-4en	Name (LAST, First)	HASEBE, Takayuki
III-4-5ja	あて名:	211-8588 日本国 神奈川県 川崎市 中原区上小田中4丁目1番１号 富士通株式会社内
III-4-5en	Address:	c/o FUJITSU LIMITED, 1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8588 Japan
III-5	その他の出願人又は発明者	
III-5-1	この欄に記載した者は	発明者である (inventor only)
III-5-4ja	氏名(姓名)	前田 直昭
III-5-4en	Name (LAST, First)	MAEDA, Naoaki
III-5-5ja	あて名:	211-8588 日本国 神奈川県 川崎市 中原区上小田中4丁目1番１号 富士通株式会社内
III-5-5en	Address:	c/o FUJITSU LIMITED, 1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8588 Japan

特許協力条約に基づく国際出願願書 0251475/2555
原本（出願用）- 印刷日時 2002年07月09日 (09.07.2002) 火曜日 14時42分35秒

III-6	その他の出願人又は発明者	
III-6-1	この欄に記載した者は	発明者である (inventor only)
III-6-4ja	氏名(姓名)	須賀 敦浩
III-6-4en	Name (LAST, First)	SUGA, Atsuhiro
III-6-5ja	あて名:	211-8588 日本国 神奈川県 川崎市 中原区上小田中4丁目1番１号 富士通株式会社内
III-6-5en	Address:	c/o FUJITSU LIMITED, 1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8588 Japan
IV-1	代理人又は共通の代表者、通知のあて名 下記の者は国際機関において右記のごとく出願人のために行動する。	代理人 (agent)
IV-1-1ja	氏名(姓名)	大菅 義之
IV-1-1en	Name (LAST, First)	OSUGA, Yoshiyuki
IV-1-2ja	あて名:	102-0084 日本国 東京都 千代田区 二番町８番地２０ 二番町ビル３F
IV-1-2en	Address:	3rd Fl., Nibancho Bldg., 8-20, Nibancho, Chiyoda-ku, Tokyo 102-0084 Japan
IV-1-3	電話番号	03-3238-0031
IV-1-4	ファクシミリ番号	03-3238-0034
IV-1-5	電子メール	osugapat@osuga-pat.com
V	国の指定	
V-1	広域特許 (他の種類の保護又は取扱いを求める場合には括弧内に記載する。)	AP: GH GM KE LS MW MZ SD SL SZ TZ UG ZM ZW 及びハラレプロトコルと特許協力条約の締約国である他の国 EA: AM AZ BY KG KZ MD RU TJ TM 及びユーラシア特許条約と特許協力条約の締約国である他の国 EP: AT BE BG CH&LI CY CZ DE DK EE ES FI FR GB GR IE IT LU MC NL PT SE SK TR 及びヨーロッパ特許条約と特許協力条約の締約国である他の国 OA: BF BJ CF CG CI CM GA GN GQ GW ML MR NE SN TD TG 及びアフリカ知的所有権機構と特許協力条約の締約国である他の国
V-2	国内特許 (他の種類の保護又は取扱いを求める場合には括弧内に記載する。)	AE AG AL AM AT AU AZ BA BB BG BR BY BZ CA CH&LI CN CO CR CU CZ DE DK DM DZ EC EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MA MD MG MK MN MW MX MZ NO NZ OM PH PL PT RO RU SD SE SG SI SK SL TJ TM TN TR TT TZ UA UG UZ VN YU ZA ZM ZW

特許協力条約に基づく国際出願願書　　0251475/2555

原本（出願用）～印刷日時 2002年07月09日 (09.07.2002) 火曜日 14時42分35秒

V-5	指定の確認の宣言 出願人は、上記の指定に加えて、規則4.9(b)の規定に基づき、特許協力条約のもとで認められる他の全ての国の指定を行う。ただし、V-6欄に示した国の指定を除く。出願人は、これらの追加される指定が確認を条件としていること、並びに優先日から15月が経過する前にその確認がなされない指定は、この期間の経過時に、出願人によって取り下げられたものとみなされることを宣言する。		
V-6	指定の確認から除かれる国	なし (NONE)	
VI	優先権主張	なし (NONE)	
VII-1	特定された国際調査機関(ISA)	日本国特許庁 (ISA/JP)	
VIII	申立て	申立て数	
VIII-1	発明者の特定に関する申立て	-	
VIII-2	出願し及び特許を与えられる国際出願日における出願人の資格に関する申立て	-	
VIII-3	先の出願の優先権を主張する国際出願日における出願人の資格に関する申立て	-	
VIII-4	発明者である旨の申立て（米国を指定国とする場合）	-	
VIII-5	不利にならない開示又は新規性喪失の例外に関する申立て	-	
IX	照合欄	用紙の枚数	添付された電子データ
IX-1	願書（申立てを含む）	5	-
IX-2	明細書	97	-
IX-3	請求の範囲	11	-
IX-4	要約	1	EZABST00.TXT
IX-5	図面	52	-
IX-7	合計	166	
	添付書類	添付	添付された電子データ
IX-8	手数料計算用紙	✓	-
IX-11	包括委任状の写し	✓	-
IX-17	PCT-EASYディスク	-	ﾌﾚｷｼﾌﾞﾙﾃﾞｨｽｸ
IX-18	その他	納付する手数料に相当する特許印紙を貼付した書面	-
IX-18	その他	国際事務局の口座への振込みを証明する書面	-
IX-19	要約書とともに提示する図の番号	3	
IX-20	国際出願の使用言語名：	日本語	
X-1	提出者の記名押印		
X-1-1	氏名(姓名)	大菅 義之	

受理官庁記入欄

10-1	国際出願として提出された書類の実際の受理の日	09.07.02

特許協力条約に基づく国際出願願書

原本（出願用）- 印刷日時 2002年07月09日 (09.07.2002) 火曜日 14時42分35秒

10-2 10-2-1 10-2-2	図面 : 受理された 不足図面がある	
10-3	国際出願として提出された書類を補完する書類又は図面であってその後期間内に提出されたものの実際の受理の日（訂正日）	
10-4	特許協力条約第11条(2)に基づく必要な補完の期間内の受理の日	
10-5	出願人により特定された国際調査機関	ISA/JP
10-6	調査手数料未払いにつき、国際調査機関に調査用写しを送付していない	

国際事務局記入欄

11-1	記録原本の受理の日	

明細書

開放型汎用耐攻撃ＣＰＵ及びその応用システム

技術分野

本発明は、機器内部の情報を改ざんしたり、秘密情報を取り出したりしようとする攻撃に対して対抗する技術に関する。

背景技術

近年、デジタルコンテンツの流通の促進に伴い、著作権などのコンテンツに係わる権利を保護する技術（ＤＲＭ：（Digital Rights Management）が提供されている。ＤＲＭの例として、例えば、三洋電機、日立、富士通の３社が共同開発したＵＤＡＣ（Universal Distribution with Access Control）が挙げられる。

ＵＤＡＣのようなＤＲＭを実装する際、コンテンツ保護のセキュリティを十分なレベルにするためには、利用者のシステムにおいてＤＲＭをＴＲＭ（Tamper Resistant Module）とすることが重要である。（以下、ＤＲＭをＴＲＭとすることをＤＲＭのＴＲＭ化という。）　現在、コンシューマ向けの製品で一般に実施されるＴＲＭ化には、大きく分けてハードウェアＴＲＭ及びソフトウェアＴＲＭの２つの方法がある。

ハードウェアＴＲＭとは、半導体の外部端子から秘密情報の読み出しなどができない構造にし、特殊コーティングや極微細化などを施すことによって実現されたＴＲＭである。ハードウェアＴＲＭをソフトウェアＴＲＭと比較して、特に「攻撃対抗容器」と呼ぶこともできる。ソフトウェアＴＲＭとは、暗号化させたいコードを解析困難な構造にして暗号化し、そのコードを実行の瞬間に

復号することによって実現されたＴＲＭである。

しかし、従来のＴＲＭには、以下のような問題があった。

１．　ハードウェアＴＲＭの問題

・コンシューマ向け用途ではコストを重視する関係上ＤＲＭを半導体パッケージ化する必要があるが、一つのチップに盛り込むことができるリソースには限りがあるため、多様な機能を提供することはできない。ライセンスキーやＣＲＬ（Certificate Revocation List）を記録する領域の大きさを拡張したい場合や、コンテンツ利用条件をＸｒＭＬ（eXtensible rights Markup Language）で表現したい場合などには、ハードウェアＴＲＭでは対応できない。

・安全性を重視すると、利用するプロセッサと記録媒体をすべてハードウェアＴＲＭ内に持つ必要があるが、このため製造後に機能を拡張させることが困難になっている。また、このことからソフトウェアが利用できる記録媒体が限定されるため、リソース消費が高い一般の（保護の必要がない）ソフトウェアを同時にプロセッサに実行させることは困難である。また、暗号鍵が異なる複数の保護ソフトウェアを同時にプロセッサに実行させることも困難である。つまり、ハードウェアＴＲＭには汎用性がない。

・コンテンツの種類の追加やＤＲＭのバーションアップごとに新たなチップの開発し、新たに半導体製造工程などのラインを起こすことが必要となるため製造コストが大きくなる。

・互換性を持つ汎用ＣＰＵチップ上にそれぞれのソフトウェアを開発・搭載するだけで何種類もの機能を利用者に提供することによりスムーズなバグ修正や機能拡張も実現してきた「情報技術の逐次進化」を阻害しかねなかった。

２．　ソフトウェアＴＲＭの問題

・おおもとの秘密鍵は暗号化することができず、どのように分散などして保管してもソフトウェア解析だけで容易に秘密鍵を見つけ出すことができる。

・ハードウェアＩＣＥ（In Circuit Emulator）を用いれば実行内容をトレースすることにより、秘密鍵を始めとする各種秘密情報を容易に見つけ出すことができる。

発明の開示

本発明の目的は、上記問題を解決し、ソフトウェアＴＲＭ並みの汎用性を持ちつつも、ハードウェアＴＲＭ並みの安全性を有するＴＲＭを提供することである。

上記目的を達成するために、本発明の１態様によれば、中央演算装置において、秘密裏に隠された第１の秘密鍵と、暗号化及び復号を行う暗号手段とを備え、前記第１の秘密鍵は、公開鍵と対であり、前記暗号手段は、前記公開鍵を用いて暗号化された第１のプログラムの第１のライセンスを前記第１の秘密鍵を用いて復号することにより、前記第１のライセンスから前記第１のプログラムを復号するためのコード復号鍵を取得するように構成する。なお、コード復号鍵は、第１のプログラムを暗号化する際に使用されたコード暗号鍵と同じであってもよい。また、第１のライセンスは、第１のプログラムに埋め込まれる事としてもよいし、両者は別々に流通する事としても良い。

このように構成することにより、前記第１のプログラムを復号するための鍵を得るために必要な秘密鍵は、中央演算装置内に秘密裏に記録されているため、分散されて保管されることはない。従って、プログラムを解析する事によって秘密鍵が容易に発見されてしまうという問題を解決する事ができる。

また、上記中央演算装置は、キャッシュを更に備え、前記暗号手段は、前記第１のプログラムを構成する暗号化ブロックがメモリ領域から前記キャッシュに出力される際に、キャッシュ単位で前記暗号化ブロックを復号する、こととしてもよい。キャッシュ単位で暗号化ブロックを復号し、キャッシュに記録す

ることにより、従来より安全性を向上させることが可能となる。

さらに、また、上記中央演算装置は、ユーザが参照したり改ざんしたりすることができない耐攻撃バッファを更に備え、前記コード復号鍵は、前記耐攻撃バッファに記録されることとしてもよい。耐攻撃バッファ内に記録された鍵は、カーネルモードであってもユーザが参照したり改ざんしたりする事ができないため、コード復号鍵を安全に保持することが可能となる。

ここで、前記耐攻撃バッファは、さらに、外部出力禁止情報及びキャッシュロック情報を含み、前記外部出力禁止情報は、対応する前記耐攻撃バッファ内の情報を前記耐攻撃バッファ外に出力しても良いか否かを示し、前記キャッシュロック情報は、対応する情報をキャッシュ外に出力しても良いか否かを示し、

前記外部出力禁止情報及び前記キャッシュロック情報に基づいて、第１のプログラム及び他のプログラムとの間における前記第１のライセンスの移動は管理されることとしてもよい。これにより、前記中央演算装置に、複製不能プライベートロッカーを実現させる事が可能となる。従って、例えば、複数のプログラム間においてライセンスを移動可能とする場合であっても、再送攻撃によるライセンスの不正コピーを防止することが可能となる。

また、上記中央演算装置において、前記第１のライセンスは、前記第１のプログラムの実行プロセスがメモリ領域にアクセスする際のアクセス条件を含み、前記第１のプログラムを構成する暗号化ブロックが記録される前記メモリ領域のアドレスと前記メモリ領域へのアクセス条件とを記録するＴＬＢ（Translation Lookaside Buffer）と、メモリ管理手段と、キャッシュと、プロセッサコアを更に備えるように構成してもよい。この構成において、前記耐攻撃バッファは前記ＴＬＢとリンクされ、前記メモリ管理手段は、前記暗号化ブロックを記録するメモリ領域のアドレスに基づいて前記ＴＬＢから前記メモリ領域へのアクセス条件を取得し、さらに、前記耐攻撃バッファから、前記メ

モリ領域に対応する前記コード復号鍵を取得する。前記プロセッサコアは、前記メモリ管理手段が取得した前記アクセス条件に基づいて、前記実行プロセスから前記メモリ領域へのアクセスを実行することが許可されているか否か判定し、前記メモリ領域へのアクセスを実行する事が許可されていると判定した場合、前記実行プロセスから前記メモリ領域へのアクセスを実行する。前記暗号手段は、前記メモリ管理手段が取得した前記コード復号鍵を用いて前記メモリ領域内の前記暗号化ブロックを復号して得られたコードを前記キャッシュに書き込む。

メモリ領域は、ＴＬＢを介して耐攻撃バッファと対応付けられており、そのメモリ領域に書きこまれたブロックを復号するための鍵は、この対応関係に基づいて取得される。さらに、メモリ領域へのアクセスも、ＴＬＢ内のアクセス条件に従って制御される。従って、安全にメモリ領域へのアクセス制御及びブロックのキャッシュへのロードを行うことが可能となる。

ここで、前記第１のプログラムの実行プロセスからアクセスされるメモリ領域が、第１のメモリ領域から第２のメモリ領域に切り替わった場合、前記メモリ管理手段は、さらに、前記耐攻撃バッファから取得された前記第１のメモリ領域に対応するコード復号鍵と、前記第２のメモリ領域に対応するコード復号鍵とが一致するか否か判定し、一致すると判定した場合、前記実行プロセスから前記第２のメモリ領域へのアクセスを実行し、一致しないと判定した場合、前記実行プロセスから前記第２のメモリ領域へのアクセスを実行しない、こととしてもよい。これにより、Call、Jump または Return 等の命令によって、あるメモリ領域から、そのメモリ領域に対応するコード復号鍵と異なる他のコード復号鍵が対応している他のメモリ領域に実行中アドレスが移動した際には、その実行プロセスから上記他のメモリ領域にはアクセスできないこととなる。これにより、あるオーナープロセスが、他のオーナープロセスのメモリ領域へ

アクセスすることを禁止することが可能となる。

さらにまた、上記中央演算装置において、前記耐攻撃バッファには前記コード暗号鍵ごとに、異なるデータ暗号鍵が記録され、前記暗号手段は、前記データ暗号鍵を用いて前記キャッシュ内のデータを暗号化した後、前記ＴＬＢを介して前記データ暗号鍵に対応付けられたメモリ領域に記録し、前記メモリ領域から暗号化されたデータを読み出して前記データ暗号鍵を用いて復号した後、前記キャッシュに書き込むこととしてもよい。これにより、データをメモリ領域に退避させたり、キャッシュにロードさせたりする操作を、安全に行う事が可能となる。

また、上記中央演算装置において、第１のコードを実行することにより得られた作業データを第２のコードで利用する場合、前記プロセッサコアは、前記作業データを記録するメモリ領域へのアクセス権を前記第２のコードに与えるように前記ＴＬＢを設定し、且つ、前記作業データを暗号化するためのコード暗号鍵を、前記第２のコードが前記作業データを前記メモリ領域から読み出す際に使用するように前記耐攻撃バッファを設定することとしてもよい。

これにより、通常、第１のコードのコード暗号鍵と、第２のコードのコード暗号鍵が異なる場合であっても、第１のコードと第２のコードとがデータ暗号鍵を共用することにより第１のコードを実行した結果得たデータを第２のコードが読み出す事が可能となる。従って、前記中央演算装置が２以上のソフトウェアを保護しながら実行する場合であっても、必要に応じて安全にメモリ領域を共有する事が可能となる。

また、上記中央演算装置において、レジスタと、レジスタへのアクセス制御を行うためのレジスタアクセス制御テーブルと、を更に備え、前記プロセッサコアは、前記レジスタアクセス制御テーブル内の封印フラグを用いて、前記レジスタの封印及び解放を制御する、こととしてもよい。これにより、プロセス

図３４は、ＤＲＭソフトウェアモジュールの構築モデルの一例を示す図である。

図３５は、メディアＤＲＭによって行われる処理を示すフローチャート（その１）である。

図３６は、メディアＤＲＭによって行われる処理を示すフローチャート（その２）である。

図３７は、デコーダＤＲＭによって行われる処理を示すフローチャート（その１）である。

図３８は、デコーダＤＲＭによって行われる処理を示すフローチャート（その２）である。

図３９は、デコーダＤＲＭによって行われる処理を示すフローチャート（その３）である。

図４０は、再生アプリケーションによって行われる処理を示すフローチャートである。

図４１は、秘密情報を維持・管理する方式を示す図である。

図４２は、ライセンス管理のためのメモリアクセスの方式を示す図である。

図４３は、第２実施形態に係わるＧＴを実現するＣＰＵの構成図である。

図４４は、第２実施形態におけるブロックの構造を示す図である。

図４５は、ebimが４である場合のブロックの構造を示す図である。

図４６は、保護ブロックセレクタによって行われる処理を示すフローチャートである。

図４７は、ハッシュエンジンによって行われる処理を示すフローチャートである。

図４８は、ハッシュチェッカによって行われる処理を示すフローチャートである。

図４９は、ＮＯＰの処理について説明する図である。

図５０は、マルチＣＰＵの構成図である。

図５１は、コードオブジェクトから保護コード実行形式を出力するツール群の例を示す図である。

図５２は、ＧＴを、パーソナルコンピュータ又はワークステーションに実装した場合のシステム構成例を示す図である。

発明を実施するための最良の形態

本発明の実施形態について図面を参照しながら説明する。なお、同じ装置等には同じ参照番号をつけ、説明を省略する。

本発明は、ＣＰＵに汎用ＴＲＭ機能を実装する事により、ＣＰＵを、汎用性を有する攻撃対抗容器とする。以下の説明において、本発明に係わる汎用ＴＲＭ機能を実装するＣＰＵを「Generic TRM」といい、ＧＴと略称する。

＜構成＞

ＧＴの構成について説明する前に、ＧＴと、そのＧＴによって実行されるソフトウェアの利用関係について説明する。図１に、ＧＴを実現するＣＰＵパッケージとそのＣＰＵパッケージによって実行されるＧＴ基本パッケージソフトウェアの利用関係モデルを示す。

図１に示すように、ＧＴ基本パッケージソフトウェアは、アプリケーション層、ＤＲＭ層、ＰＫＩライブラリ層に分けることができる。アプリケーション層には、保護されるアプリケーション（以下、保護アプリケーションという）が含まれる。ＤＲＭ層には、デコーダＤＲＭ３０及びメディアＤＲＭ４０が含まれる。ＰＫＩライブラリ層には、ＰＫＩライブラリ２０が含まれる。デコーダＤＲＭ３０、メディアＤＲＭ４０及びＰＫＩライブラリ２０は、ＯＳ（Operation System）の一部である。

各基本ソフトウェアパッケージはそれぞれ次のような機能を持つ。なお、これらの機能は、従来から知られているため説明は省略する。

ＰＫＩライブラリ：

・標準のPKIX (Public Key Infrastructure (X.509))ライブラリ

メディアＤＲＭ (Virtual Media DRM, Trusted Server DRMを含む)：

・ライセンス生成・管理・削除・移動・複製の制御

・保護コンテンツ管理

・UDAC-MB/LB/PI認証・ライセンス管理機能

・ライセンス変換機能：MB (Media Base)、PI (Protocol Independent)、GT間での変換など

デコーダＤＲＭ：

・メディアＤＲＭからのライセンス取得

・許諾ライセンス復号とコンテンツ復号鍵の維持

・コンテンツの復号、汎用利用制御とアプリケーションへの安全な転送

・利用制御機能拡張機能

図１に示すように、各ＤＲＭ３０及び４０は、ソフトウェアとして実現され、ハードウェアとして実現されるのではない。また、ＧＴ１０はＣＰＵとして実現される。ＧＴ１０（ＣＰＵ）は、ＤＲＭ３０及び４０、ＰＫＩライブラリ２０及び保護アプリケーション５０を実行する。

図２に、図１に示す基本パッケージ以外のモジュールを含めたＧＴ上でのプログラミングモデルを示す。

図２に示すように、ＯＳには、上述のＰＫＩライブラリ２０、デコーダＤＲＭ３０及びメディアＤＲＭ４０に加えて、ＯＳカーネル６０及びデバイスドライバ７０を備える。デバイスドライバ７０は、周辺ハードウェア６０を動作させるために必要なソフトウェアである。

保護アプリケーション５０及び保護されないアプリケーションを含む複数のアプリケーションは、このＯＳ上で動作する。つまり、ＧＴ１０上では、ＤＲＭ機能を用いた保護アプリケーションとともに、従来のＣＰＵ上で稼動する他のアプリケーションも稼動する。このように、ＧＴ１０は、一般の保護の必要がないアプリケーションを保護アプリケーションと同時に実行することが可能な汎用性を有する耐攻撃容器である。

なお、ＯＳカーネル６０は、従来の動作に加え、ＧＴ１０において、メモリ管理及びコンテキストの切り替え制御の際に、レジスタ封印等のＧＴ１０に固有の処理（後述）を行う。しかし、それらの処理をＯＳカーネル６０に行わせることは、ＧＴ１０のセキュリティに影響を及ぼさない。すなわち、ＯＳカーネル６０にセキュリティホールが存在している場合であっても、ＧＴ１０による保護の安全性には影響はない。

デコーダＤＲＭ３０、メディアＤＲＭ４０及びこれらのＤＲＭが用いるＰＫＩライブラリ２０内の暗号・復号ライブラリ、更には、ＴＲＭ化が必要なアプリケーションは、ＧＴ１０上で保護されるＧＴ保護コードとして流通し、実行される必要がある。そして、これらのソフトウェアモジュールは、ほぼ全文を暗文にされる必要がある。

上述のように、従来、ソフトウェアＴＲＭには、拡張性はあるが容易に破壊できるという性質があった。しかし、本発明によれば、ＧＴ１０によって保護されるＤＲＭソフトウェアモジュールを採用する事により、ＤＲＭソフトウェアモジュールにハードウェアＴＲＭ並みの強度を与える事を可能としている。一方で、ハードウェアＴＲＭには拡張性が無く、リソースも限られるという問題があったが、ＧＴ１０によれば、ＤＲＭソフトウェアを採用しているため、機能的な拡張には、ＧＴ１０を搭載するＣＰＵに変更を加える必要は無く、ＤＲＭソフトウェアのバージョンアップによって対応する事が可能である。

ＤＲＭ構築モデルは、ＵＤＡＣ－ＭＢ (Universal Distribution with Access Control-Media Base)ＵＤＡＣ－ＬＡ、ＵＤＡＣ－ＰＩそれぞれのアーキテクチャ及び仕様に従うこととしてもよい。

以下、図３を用いて、第１実施形態に係わるＧＴを実現するＣＰＵの構成について、データのやり取りについて言及しながら説明する。

図３に、平文又は暗文のみからなるコードブロック或いはデータブロックを想定した場合の（ebim=0, 1 のみをサポートする場合）、ＧＴ内部の回路ブロックの構成と回路ブロック間のデータ交換モデルを示す。

ＧＴ１０は、プロセッサコア１１、命令キャッシュ１２、データキャッシュ１３、命令MMU (Memory Management Unit) １４、データMMU１５、暗号エンジン１６を備え、ＲＡＭ (Random Access Memory) １７に接続されている。

本発明に係わるＧＴ１０の特徴の１つとして、コードブロック又はデータブロックを暗号化したり、復号したりする暗号エンジン１６を備え、この暗号エンジン１６を用いて暗号化されたコードブロックやデータブロックを復号してからＧＴ１０内部のキャッシュ１２又は１３に保持し、データキャッシュ１２からＲＡＭ１７に作業データを出力する際には、暗号エンジン１６を用いてそのデータを暗号化してから出力することがある。以下、ＧＴ１０におけるブロック間のデータ交換について説明する。

まず、ＲＡＭ１７から入力されたコードブロックは、プロセッサコア１１及び命令MMU１４によって保護すべきコードブロック（以下、保護コードブロックという）又は平文コードブロックのいずれであるか識別される。ＲＡＭ１７から入力されたデータブロックは、プロセッサコア１１及びデータMMU１５によって保護すべきデータブロック（以下、保護データブロックという）又は平文データブロックのいずれであるか識別される。

保護コードブロック及び保護データブロックは暗号エンジン１６に送信され

るよう、それぞれ命令MMU１４及びデータMMU１５からＲＡＭ１７に対して保護ブロック転送先のアドレスの指定がされる。暗号エンジン１６に出力されたコードブロックやデータブロックは復号されて、それぞれ命令キャッシュ１２やデータキャッシュ１３に出力される。保護コードブロックを復号する際に用いられる鍵 Kc 及び保護データブロックを復号する際に用いられる鍵 Kd は、プロセッサコア１１から暗号エンジン１６に出力される。平文コードブロック又は平文データブロックは、そのまま、それぞれ命令キャッシュ１４又はデータキャッシュ１５に出力される。

また、保護コードブロック及び保護データブロックの安全性を確保するために、ＧＴ１０によれば、作業データをデータキャッシュ１３からＲＡＭ１７に出力する際に、データキャッシュ１３の出口において作業データを暗号化する。そのために、まず、データキャッシュ１３から作業データが暗号エンジン１６に出力されるとともに、プロセッサコア１１から作業データを暗号化するための鍵 Kd（保護データブロックを復号する際に用いる鍵と同じ）が暗号エンジン１６に出力される。暗号エンジン１６は、その鍵 Kd を用いてその作業データを暗号化し、暗号化された作業データをＲＡＭ１７等に設けられた仮想メモリ領域に出力する。さらに、暗号化する際に、データブロックに乱数を付加することとしても良い。

なお、データブロックを仮想メモリ領域に出力する際には、そのデータブロックが保護されるべきデータブロックであるか否か、ＴＬＢ（Translation Look aside Buffer）（不図示、後述）に基づいてプロセッサコア１１によって判断され、判断結果に基づいて、データキャッシュ機能からのアドレス指定の際に暗号エンジン１６を通すかどうかが決定される。また、暗号化及び暗号の復号に用いる鍵 Kc 及び Kd は、ＧＴ１０内の暗号エンジン１６がＧＴライセンスを復号することによって得られ、耐攻撃バッファ（以下、ＴＲＢ：Tamper

Resistant Buffer という）（不図示、後述）に格納されている。

命令MMU１４とデータMMU１５は、図３において、別々のブロックとして示されているが、一つのモジュールにしてもよい。命令キャッシュ１２とデータキャッシュ１３についても同様である。

上記のように、ＧＴ１０によれば、キャッシュの出入り口で暗号化されたコードブロックやデータブロックをキャッシュ単位で復号する。ところで、ＣＰＵ内でコードを１つずつ暗号化する場合、２バイト程度の単位で行う必要がある。しかし、十分な強度を得るためには現在の技術では８バイト程度を暗号化ブロックとすることが要求されるため、２バイト程度の単位では十分な強度が維持できない。しかし、ＧＴ１０によれば、ＲＡＭ１７からの暗号化されたコードブロックやデータブロックを復号する際にはキャッシュ単位で復号して維持する事により、保護コードブロック及び保護データブロックを安全に守る事が可能となる。

なお、ＧＴ１０に命令キャッシュ１２が備えられない場合、ＧＴ１０内にＲＡＭ領域を設け、全ての実行コードを内部で復号して内部のＲＡＭ領域に保存してから実行することとしてもよい。これにより、上記構成と同等の安全性を確保する事ができる。また、ＧＴ１０にデータキャッシュ１３が備えられない場合、ＧＴ１０の内部にＲＡＭ領域を設け、保護が必要な作業データをＧＴ１０の内部のＲＡＭ領域に記録する。これにより、上記構成と同等の安全性を確保する事ができる。

また、従来、ＣＰＵ内の作業データを一時的にＲＡＭに出力する際に、秘密にすべきデータが漏洩することがあった。また、そのデータの内容を監視することによって命令コードの推測を行うことによって、ＣＰＵ内に保持されている秘密鍵を推測することも可能であった。しかし、ＧＴ１０によれば、データキャッシュから作業データをＲＡＭに出力する際に暗号化し、キャッシュに復

がデータをレジスタに格納する場合には、現在実行中のプロセス以外からレジスタに格納されたデータにアクセスする事ができないように封印することが可能となる。

また、上記中央演算装置において、前記ＴＬＢの内容を外部のページテーブルに記録する際には、前記暗号手段は、記録する内容に署名を付与し、前記ページテーブルの内容を前記ＴＬＢに取り込む前には、前記暗号化手段は、前記署名が正しいことを確認する、こととしてもよい。これにより、ＴＬＢの内容を安全にページテーブルに保持させる事が可能となる。

また、上記中央演算装置において、前記耐攻撃バッファの内容を外部記憶装置内の暗号化キーテーブルに記録する際には、前記暗号手段は、記録する内容を暗号化することとしてもよい。これにより、耐攻撃バッファの内容を外部記憶装置に安全に保持する事が可能となる。

なお、ＴＬＢの内容の署名を作成する際に使用される秘密鍵、及び耐攻撃バッファの内容を暗号化する際に使用される秘密鍵は、ともに、中央演算装置内のプロセッサによって生成されることとしてもよい。

また、前記中央演算装置は、他の中央演算装置と接続され、前記中央演算装置は、前記他の中央演算装置と相互に認証を行う事にことによりセッション鍵を取得し、前記中央演算装置の前記暗号手段は、前記セッション鍵を用いて前記中央演算装置の前記キャッシュの内容を暗号化して、前記他の中央演算装置に同期転送する、こととしてもよい。これにより、上記構成を有する中央演算装置を複数有するマルチＣＰＵにおいて、キャッシュ内の内容を安全に同期させることが可能となる。

また、上記中央演算装置において、前記暗号手段は、前記第１のプログラムを実行する前に、前記公開鍵を用いて第２のプログラムに付加された前記第２のライセンスを復号する事により第２の秘密鍵を暗号化する際に用いられた秘

密鍵暗号鍵を取得し、さらに、前記取得された秘密鍵暗号鍵を用いて前記第２の秘密鍵を復号する、こととしてもよい。

また、この際に、前記第２のライセンスには、第１のプログラムの実行プロセスから読み出しのみ可であることを示すアクセス条件が付加されており、前記第２の秘密鍵は、前記第１のプログラムの実行プロセスからの読み出しのみ可能であることとしてもよい。第２の秘密鍵を第１のプログラムの実行プロセスからのみ読むことができないこととすることにより、秘密情報を安全に維持管理することが可能となる。

なお、前記第１のプログラムは、ＴＲＭ化が要求されるどのようなソフトウェアであってよい。例えば、第１のプログラムとして、トラステッドコンピューティングモジュール、前記中央演算装置に電子財布を実現させるプログラム、個人情報を扱うソフトウェア、前記中央演算装置の実装コードのウィルスチェックソフトウェア、複数の中央演算装置間を移動する移動エージェント、ロボットの制御プログラム、ＩＣカード用のセキュリティプログラム等が挙げられる。

また、上記中央演算装置において、前記第１のプログラムを構成するブロックには様々な形態が考えられ、それに応じて中央演算装置に更なる手段を備える事としてもよい。

例えば、ブロックは、ブロックのハッシュ値の確認が必要であるか否かを示すハッシュ確認要否情報を含み、前記ハッシュ確認要否情報に基づいて、前記ブロックのハッシュ値を算出し、前記ブロックに付加するハッシュ手段と、前記ハッシュ確認要否情報に基づいて、前記ブロックの前記ハッシュ値を確認するハッシュ確認手段と、を更に備えることとしてもよい。

また、例えば、ブロックは、ブロックが保護を必要とするか否かを示す暗号化要否情報を含み、前記暗号化要否情報に基づいて、前記ブロックを前記暗号

手段に出力するか、そのままキャッシュ又はメモリ領域に出力するか判定する保護ブロック選択手段を、更に備えることとしてもよい。

また、ブロックごとにブロックを選択する保護ブロック選択手段の付加を低減するために、以下のようにしてもよい。

例えば、第１のプログラムの実行ファイルのヘッダは、前記第１のプログラムを構成するブロックの構成を示す暗号化ブロックビットマップを含み、保護ブロック選択手段は、前記暗号化ブロックビットマップに基づいて、前記ブロックを前記暗号手段に出力するか、そのままキャッシュ又はメモリ領域に出力するか判定する。

また、例えば、第１のプログラムのコードの先頭は、前記第１のプログラムを構成する複数のブロックが、平文ブロックと暗号化ブロックの組合せの繰り返しであり、前記組合せにおいて平文ブロックが連続する数及び暗号化ブロックが連続する数を指定するコードであり、

プロセッサコアこのコードを実行することにより、前記ブロックを前記暗号手段に出力するか、そのままキャッシュ又はメモリ領域に出力するか判定する。

また、上記中央演算装置において、前記キャッシュとメモリの間に、前記暗号手段を介するキャッシュラインと、前記暗号手段を介さないキャッシュラインと、を更に備えることとしてもよい。これれにより、処理の高速化を図る事ができる。

また、本発明の他の態様として、中央演算装置に保護プログラムを実行する許諾を与える制御を行わせるプログラムであって、前記保護プログラムは、コード暗号鍵によって暗号化され、保護プログラムに対応して、前記コード暗号鍵を含み、且つ、前記中央演算装置に秘密裏に備えられた秘密鍵と対になる公開鍵によって暗号化されたライセンスが存在し、前記中央演算装置が前記保護プログラムを実行する前に、前記ライセンスを前記中央演算装置に投入し、前

記中央演算装置に備えられた暗号手段に、前記秘密鍵を用いて前記ライセンスを復号することにより、前記ライセンスから前記コード暗号鍵を取得させ、前記暗号手段に、前記コード暗号鍵を用いて前記保護プログラムを復号させる、ことを含む処理を前記中央演算処理装置に実行させる、ように構成する。

このプログラムは、上記構成を有する中央演算装置によって行われる処理と同様の作用・効果を得ることができるものである。従って、このプログラムによっても、上記問題を解決する事ができる。また、上記プログラムを記録する記録装置も、上記プログラムを上記中央演算装置によって実行されることにより、上記構成を有する中央演算装置によって行われる処理と同様の作用・効果を得ることができるものである。

また、上記中央演算装置を構成する各手段によって行われる動作を手順として含むソフトウェア実行許諾方法も、上記構成を有する中央演算装置によって行われる処理と同様の作用・効果を得ることができるものである。

また、本発明の更なる１態様によれば、コンピュータにおいて実行されるプログラムであって、コード暗号鍵によって暗号化されたプログラムと、前記保護プログラムに対応して、前記コード暗号鍵を含み、且つ、前記プログラムを実行するべきコンピュータに備えられた中央演算装置が秘密裏に備える秘密鍵と対になる公開鍵によって暗号化されたライセンスが存在し、前記ライセンスは、前記プログラムが実行される前に前記中央演算装置に投入され、前記中央演算装置によって前記秘密鍵を用いて復号され、前記プログラムは、前記ライセンスから取得された前記コード暗号鍵を用いて、前記中央演算装置によって復号される、ことを特徴とする。このプログラムは、上記中央演算装置によって保護されるソフトウェアであり、上記中央演算処理装置を有するコンピュータによって実行される事によって、ハードウェアＴＲＭ並みの安全性を得ることができる。

また、上記プログラムを記録する、前記コンピュータによって読み取り可能な記録媒体もまた、その記録媒体からプログラムをロードし、プログラムを上記中央演算処理装置を有するコンピュータによって実行される事によって、上記プログラムと同様の作用・効果を得ることができる。

また、本発明の更なる別の態様によれば、上記構成を有する中央演算装置を備えるコンピュータにおいて実行されるプログラムを生成するプログラム生成装置であって、コードオブジェクトを入力する入力手段と、入力された前記コードオブジェクトを複数のブロックに分割し、それぞれのブロックにＮＯＰ命令を追加するリンカ前処理手段と、アドレス解決を行うリンカ手段と、コード暗号鍵を用いて各ブロックを暗号化することにより保護コード実行形式を生成する保護コード実行形式生成手段と、前記コード暗号鍵を含み、前記秘密鍵と対になる公開鍵によって暗号化されたライセンスを生成するライセンス生成手段とを備え、前記ライセンスは、前記コンピュータが前記保護コード実行形式を実行する前に前記中央演算装置に投入され、前記暗号手段によって前記秘密鍵を用いて復号され、前記保護コード実行形式は、前記ライセンスから取得された前記コード暗号鍵を用いて、前記暗号手段によって復号される、ことを特徴とする。

このプログラム生成装置によって、上記中央演算装置によって保護を受けることができない形式のコードモジュールから、上記中央演算装置によって保護を受けることが可能なプログラムを生成することが可能となる。生成されたプログラムは、上記中央演算処理装置を有するコンピュータによって実行される事によって、ハードウェアＴＲＭ並みの安全性を得ることができる。

図面の簡単な説明

図１は、基本パッケージソフトウェアの利用関係モデルを示す図である。

図２は、プログラミングモデルを示す図である。

図３は、第１実施形態に係わるＧＴを実現するＣＰＵの構成図である。

図４は、ＴＲＢ内の各行のフィールドの構成を示す図である。

図５は、ＴＬＢ内の各行の拡張フィールドの構成を示す図である。

図６は、ＦＲアーキテクチャの場合について、ＴＬＢ内のフィールドの値とアクセス権限の対応を示す図である。

図７は、インテルアーキテクチャの場合について、ＴＬＢ内のフィールドの値とアクセス権限の対応を示す図である。

図８は、暗号化キーテーブルの構造を示す図である。

図９は、署名方式の一例を示す図である。

図１０は、ＴＲＢ、ＴＬＢ、暗号化キーテーブル及びページテーブルの関係を示す図である。

図１１は、ＴＲＳＳフラグを格納するフィールドの構成の一例を示す図である。

図１２は、ＲＡＣＴの各行のフィールド構成の一例を示す図である。

図１３は、ＲＳＳＬレジスタのフィールド構成の一例を示す図である。

図１４は、一般の仮想セグメント空間と耐攻撃セグメント空間との間でコード実行プロセスからデータにアクセスする際のポリシーを示す図である。

図１５は、ＧＴ１０上で保護プロセスが動作する様子を示す図である。

図１６は、ＧＴの認証を説明する図である。

図１７は、ＧＴライセンスに設定可能なアクセス権と被許諾権限を示す図である。

図１８は、上述のＧＴ認証処理の一部を示すフローチャートである。

図１９は、超流通ファイル形式として用いる場合の保護コード実行形式の一例を示す図である。

図２０は、保護コードのロードと実行手順、及び保護データの退避と維持について説明する図である。

図２１は、保護コードを実行する際の保護コードを記録するページへのアクセス制御について説明する図である。

図２２は、プロセッサコアによって行われる保護コード及び保護データへのアクセス制御を示すフローチャートである。

図２３は、例外・割り込み処理にかかわるフローチャートである。

図２４は、命令処理にかかわるフローチャートである。

図２５は、保護コードファイルを起動する際における、ＯＳカーネル６０及びＧＴの動作について説明する図である。

図２６は、保護コードを実行するプロセスから保護データをアクセスするメカニズムについて説明する図である。

図２７は、親プロセスから他の保護コードモジュールを呼び出す際に行う手順について説明する図である。

図２８は、親プロセスから他の保護コードモジュールを呼び出す際にＯＳカーネルが行う処理のフローチャートを示す。

図２９は、耐攻撃コード復帰例外処理による封印レジスタ不正アクセス防止のためのシーケンス例を示す図である。

図３０は、ＯＳカーネルによる保護コンテキスト切り替え時のシーケンスの一例を示す図である。

図３１は、保護データ領域の共有の一例を示す図である。

図３２は、モジュールの認証、並びに、保護データ復号鍵共有手順の設定手順について説明する図である。

図３３は、耐攻撃領域共有システムコールを呼び出した際の処理を示すフローチャートである。

活させる時にそのデータを復号する事としているため、このような問題を回避する事が可能となる。

なお、キャッシュ１２及び１３とＲＡＭ１７の間に、平文のコードブロック用のキャッシュライン、平文のデータブロック用のキャッシュライン、保護コードブロックを復号するためのキャッシュライン、保護データブロックを暗号化するためのキャッシュライン及び、保護データブロックを復号するためのキャッシュラインのように、複数のキャッシュライン並列に備える事としても良い。これにより、ＧＴ１０による処理を高速化し、保護コードブロック及び保護データブロックの安全性を高めることが可能となる。

また、図３に示すように、プロセッサコア１１内には、レジスタが備えられる。　ＧＴ１０によれば、従来の仮想メモリ領域に、耐攻撃領域という機構を追加し、この耐攻撃領域内では、安全上のリスクが互いに影響することなく、複数の保護ソフトウェアを実行する事を可能としている。そのために、プロセッサコア１１は、耐攻撃セグメントセレクタ（以下、ＴＲＳＳという）フラグ、レジスタアクセスコントロールテーブル(以下、ＲＡＣＴという）及びレジスタ封印状態リストレジスタ（以下、ＲＳＳＬレジスタという）を備える(不図示)。

ＴＲＳＳフラグは、仮想メモリ領域内のセグメントを指定するセグメント指定レジスタ内のフィールドに記録される。プロセッサコア１１は、ＴＲＳＳフラグに基づいて、レジスタによって示される仮想アドレスが、耐攻撃セグメント空間内であるのか、一般の仮想セグメント空間内であるのかを判断する事ができる。また、複数の保護ソフトウェアを実行する際に、現在実行中のプロセス以外のプロセスからレジスタにアクセスすることができないように、プロセッサコア１１は、ＲＡＣＴを用いて、レジスタの封印及び解放を制御する。また、さらに、レジスタが封印されているのか解放されているのかを示す情報は、

ＲＳＳＬレジスタに登録される。耐攻撃セグメント空間及び耐攻撃領域並びにレジスタの封印及び解放について詳しくは後述する。

このように、ＧＴ１０は、ＴＲＭ化した１ロットのＣＰＵを用いて実現される。そして、ＧＴ１０は、ＧＴ１０上で動作する、ＤＲＭソフトウェアモジュール又はその他の保護を要するモジュールをＴＲＭ化することができる。従って、ハードウェアＴＲＭを製造するコスト、特に初期ロットの開発コストを削減する事が可能となる。

以下、ＴＲＢ、ＴＬＢ、ＴＲＳＳレジスタ、ＲＡＣＴ及びＲＳＳＬレジスタについて順に説明する。

ＴＲＢは、プログラムコード（インストラクションの連結）を復号するための鍵 Kc 及びデータを暗号化／復号化する鍵 Kd 等を保持する。なお、Kd 及び Kc を合わせてソフトウェア暗号鍵という事もある。ＴＲＢは、カーネルモードであってもユーザが内部を参照・改ざんすることができない構造、すなわち耐攻撃構造をもつ。ＴＲＢ内で Kc や Kd を保持する位置の識別は Key ID によって行われ、ＴＬＢ内にもリンクした鍵の位置を識別する Key ID を持つ。この Key ID を用いて、ＴＲＢとＴＬＢをリンクさせる。

図４に、ＴＲＢ内の各行のフィールドの構成表を示す。図４の表に示すように、ＴＲＢ内の各行は、有効性フラグ p、外部出力禁止フラグ uo、キャッシュロックフラグ cl、鍵識別情報 k id、暗号鍵 key、被許諾コード鍵 ackey 及び例外処理アドレス eadr 等のフィールドを含む。また、これらのフィールドのサイズは、例としてあげたものであり、ＧＴ１０のアーキテクチャや用途によって他の最適な値にすることが可能である。

有効性フラグ p は、ＴＲＢが有効であるか無効であるかを示すフラグであり、オン（１）である場合、ＴＲＢが有効であることを示す。

外部出力禁止フラグ uo は、その行の内容を暗号化キーテーブルに出力しても

良いか否かを示すフラグであり、オン（１）である場合、暗号化キーテーブルに出力されない。但し、この場合でも、管理情報として必要な p、uo、cl 及び kid は、出力可能としてもよい。外部出力禁止フラグ uo のデフォルト値は、オフ（０）であり、その場合はその行の内容は暗号化キーテーブルに出力される。この場合、ＴＲＢ内の内容と暗号化キーテーブル内の内容とを同期させる必要がある。なお、暗号化キーテーブルは、ＧＴ１０内のＴＲＢの内容を格納する格納手段である。暗号化キーテーブルについて詳しくは後述する。

キャッシュロックフラグ cl は、鍵識別情報 kid によって指定された耐攻撃領域内のデータがキャッシュの外に出力されるか否かを示し、オン（１）である場合、そのデータ又はコードはキャッシュの外に出力されない。なお、cl がオン(１)である場合、次の２つのモードを切り替える機能を更にＧＴ１０に備える事としても良い。

（ａ）CPU 内蔵キャッシュまでしか保護データを出さないモード

（ｂ）外部（三次）キャッシュまで保護データを平文で記録するモード

（ａ）は保護データを記録する領域を少なくし、暗号・復号処理をしないことで処理性能を上げる必要がある場合にも利用することができる。（ｂ）は処理性能向上を期待できるが、保護強度のレベルは落ちることとなる。

鍵識別情報ｋid は、鍵を識別する情報であり、ＴＬＢとＴＲＢをリンクさせるために用いられる。

暗号鍵 key は、そのラインにリンクするページに書き込まれたコード又はデータを暗号化・復号するための暗号・復号鍵の値である。暗号鍵は、例えば、対称鍵（共通鍵）暗号法の鍵である。このフィールドに Kc や Kd が記録される。

被許諾コード鍵 ackey は、リンクするページにアクセス可能な保護プロセスの実行コードを復号するための復号鍵である。ここで、保護プロセスとは、保護コードが実行された状態をいう。その実行コードのページは、ＴＲＢ内の

行の鍵識別情報 kid と同じ kid を含むＴＬＢ中の行で、ページ番号を用いて指定され、そのページへのアクセス権の種類もＴＬＢ中の行で指定される。通常、被許諾コード鍵 ackey は、他の行及び自行の暗号鍵 key である。ただし、ackey の全ビットが' 0' であれば、すべてのプロセスが、ＧＴライセンスを用いてアクセス権を許諾されたことを示すものとする。また、ACgt.ackey（ACgt の ackey フィールド(後述）の値）に ackey を公開鍵 KPgt（後述）で暗号化された値が入っている場合（つまり、ACgt.aef が on の場合）には、これを復号した結果が TRB.ackey に入る。

例外処理アドレス eadr は、key が異なるページからこのＴＲＢ内の行にリンクしたページに復帰する直前に発生する例外処理コードの先頭アドレスを示す。例外処理アドレス(eadr)は、ACgt.eadr（ACgt 内の eadr フィールドの値）が含まれない AUTHORIZE 命令により TRB 内に行が設定された時点では、デフォルト値として全ビットに'0'が設定される。保護プロセスへの復帰時等に、「耐攻撃コード復帰例外アドレス不正例外」が発生した際には、そのプロセスの実行を停止し、当該 TRB の有効性フラグ(p)を off にしなければならない。なお、将来、データページごとにアクセス例外を設ける必要がある場合には、TRB.eadr（ＴＲＢ内の eadr フィールドの値）は、TRB.ackey（TRB 内の ackey フィールドの値）で暗号化されたコードの実行時に実行する例外処理コードのアドレスとして定義することもできる。

鍵 key フィールドに格納されるコード復号鍵 Kc 及びデータ暗号化・復号鍵 Kd は、例えば、乱数として生成する対称鍵暗号方式の暗号鍵であり、ＧＴ１０において、暗号・復号の前と後でデータ長が同じになる暗号方式を選択することとしてもよい。鍵を乱数生成アルコリズムを用いて生成しても良いし、ＧＴ１０内の熱乱数発生機能などを用いて生成してもよい。前者の生成法より後者の生成法の方が安全である場合が多い。Kc 及び Kd は、後述のＧＴライセンス

に含まれる。コード復号鍵 Kc 又はデータ復号鍵 Kd と、アクセス権等を埋め込んだＧＴライセンスをパラメタとしてＣＰＵインストラクション「アクセス許諾命令（AUTHORIZE 命令）」（後述）を実行する事により、耐攻撃領域内のＴＬＢ（後述）にリンクされたＴＲＢ行内の各フィールドに値が設定される。

ＴＬＢは、保護コード及び保護データを格納するアドレス並びにページへのアクセス権を管理する。図５に、ＴＬＢ内の各行の拡張フィールドの構成表を示す。図５に示すように、ＧＴ１０に係わるＴＬＢ内の各行は、従来のＴＬＢが持つフィールドに加え、有効性フラグ p、領域識別子 id、物理ページ番号 ppn、仮想ページ番号 vpn、アクセス権限 rights、鍵識別情報 kid、暗号化ブロック識別方式 ebim 及びデジタル署名 sign 等のフィールドを含む。なお、アクセス権限 rights フィールドは、権限レベル pl、アクセス権 ar、耐攻撃性フラグ tr、デバッグモードフラグ df に分けられる。これらのすべてのフィールドはＧＴ１０内ではこの耐攻撃領域内になければならない。また、図５において示されたフィールドのサイズは、例としてあげたものであり、ＧＴ１０のアーキテクチャや用途によって他の最適な値にすることが可能である。

有効性フラグｐは、ＴＬＢが有効であることを示す。

領域識別子 id は、ＴＬＢ内の当該行が示すページ領域の識別子である。

物理ページ番号 ppn は、物理ページ番号を示す。仮想ページ番号 vpn は、仮想ページ番号を示す。

アクセス権限 rights は、その当該行が示すページへのアクセス権限を示す。権限レベル pl はページにアクセス可能な権限のレベルを示し、アクセス権 ar は、ページへのアクセス権の種類を示す。権限レベル pl 及びアクセス権 ar は、図６及び図７に示すようにしてＴＬＢの各フィールドの値に基づいて決定され、ＴＬＢに設定される。なお、図６は、ＦＲアーキテクチャの場合を示し、図７は、インテルアーキテクチャの場合を示す。

耐攻撃性フラグ tr は、当該行が示すページが耐攻撃セグメント空間内（後述）内にあるか否かを示し、耐攻撃性フラグ tr がオン（１）であれば、そのページは耐攻撃セグメント空間内にあり、その行に対応する行がＴＲＢ内にあることを示す。耐攻撃性フラグのオン・オフはＧＴ１０の認証時に行われる。

デバッグモードフラグ df は、デバッグモードが有効であるのか無効であるのを示す。デバッグモードフラグ df がオン（１）であれば、有効である。耐攻撃性フラグ tr とデバッグモードフラグ df がオン（１）である場合、アクセス権 ar で指定された値に応じたデバッグモードが動作し、各 ar の値の意味は以下のようになる。

(a) ＰＲの場合、全プロセスから読み出しのみ可能：Ｒ

(b) Ｘの場合、全プロセスからの読み出しと実行が可能：ＲＸ

(c) ＰＲＷの場合、全プロセスからの読み出しと書き込みが可能：ＲＷ

(d) ＰＷＸの場合、全プロセスからの読み書きと実行が可能：ＲＷＸ

鍵識別情報 kid は、ＴＲＢ内の鍵情報にリンクするためにＴＲＢ内の行（insertion）を識別する情報である。

暗号化ブロック識別方式 ebim は、暗号化されているコードブロック又はデータブロックを識別する情報である。

ａ）ebim＝０の場合、ブロックは平文である。

ｂ）ebim＝１の場合、ブロックは暗文である。（デフォルト値であり、ebim フィールドが省略されている場合、ebim＝１として扱われる）

デジタル署名 sign は、上述の vpn から ebim までのフィールドを連結したフィールド連結データのデジタル署名であり、ＧＴ１０が生成する。

保護コード及び保護データが格納される領域を示すＴＬＢインサーション（行）内ラインのアクセス権限の値は、後述の TLB.tr がオフの場合に限り、ＯＳが管理する。従来通り、本発明でもアクセス権限の値は３ビット程度で表

現されるが、本発明によれば、さらにページが「耐攻撃領域内にあることを示すフィールドとして「耐攻撃性フラグ tr」フィールドをＴＬＢに設ける。また、この耐攻撃領域を利用可能なコード実行状態を「耐攻撃モード(Tamper Resistant Mode)」と呼ぶものとする。

保護コードおよび保護データを格納するアドレスはそのページを利用する前にＴＬＢに設定される。アクセス権限 rights 及び暗号化ブロック識別方式 ebim、被許諾コード鍵 ackey は、ＧＴライセンスに含まれるＧＴアクセス条件 ACgt（後述）に基づいて、決定される。

コード復号鍵 Kc 又はデータ暗号・復号鍵 Kd と、アクセス権とを埋め込んだＧＴライセンスをパラメタとしてＣＰＵインストラクション「アクセス許諾命令」（後述）を実行する事により、ＴＬＢ行内に、保護ページごとに KeyID が指定され、各 KeyID に対応するＴＲＢ行内の key フィールドに復号鍵が設定される。

以下、ＴＲＢ及びＴＬＢが格納される場所について説明する。

ＴＲＢは、ＧＴ１０内に格納されるが、ＧＴ１０に備えられた記憶容量が一杯になる場合が想定される。この場合、ＧＴ１０内のＴＲＢの内容の少なくとも一部を暗号化し、暗号化された内容をＲＡＭ１７やハードディスク等の外部記憶装置に記録する。この外部に記録されたＴＲＢ内の行のリストを暗号化キーテーブルという。そして、ＧＴ１０は、ＴＲＢの内容を暗号化した状態で外部記憶装置内の暗号化キーテーブルに記録しながら管理し、必要な情報を暗号化キーテーブルから取り出してＧＴ１０内で復号して利用する。ＴＲＢの内容を暗号化したり復号したりする際に用いられる鍵 Ktrb は、例えば、対称鍵暗号法の秘密鍵である。この暗号鍵 Ktrb は、ＧＴ１０の入力電源が落ちたり、ＧＴ１０がリセットされたりした場合であっても、継続して維持されることが望ましい場合がある。

以下、図８を用いて暗号化キーテーブルの構造について説明する。図８に示すように、暗号化キーテーブルには、有効性フラグ p、外部出力禁止フラグ uo、鍵識別情報 kid 及び暗号化されたＴＲＢの内容を格納するフィールド等が含まれる。

図８に示すように、暗号化キーテーブルに記録する際、各行にヘッダとして平文のままの TRB.p（ＴＲＢ中の有効性フラグ p フィールドの値）と平文のままの TRB.kid（ＴＲＢ中の鍵識別情報 kid フィールドの値）を付加する。これにより、ＯＳ（スーパーバイザモード）によって暗号化キーテーブルを管理することが可能となるが、Key ID 以外の暗号化されている内容は参照も改ざんもできないこととなる。ＴＲＢ内のフィールドのうち、p と kid とは平文の状態で特定レジスタなどからみえてもよい。

ＴＲＢの内容を外部記録装置に記録する際には、他の保護データブロックと同様にデータキャッシュ１３の暗号化ラインを通して行う。また、図８に示すように、ＴＲＢの各行を暗号化する際には行の先頭に乱数を付加してから暗号化しなければならない。これにより、Kc（コードの暗号鍵）を指定したソフトウェア開発者が故意にＴＲＢ中の暗号鍵 key フィールドの値を何度も入れなおして、平文と暗文のペアを大量に生成することにより、容易にＴＲＢを暗号化する暗号鍵Ktrb を解読することを防止する。

さらに、暗号化キーテーブルをＲＡＭ１７から不図示のハードディスクに記録しておき、ＧＴ１０の再起動後に再利用することとしてもよい。特にハードウェアＴＲＭレベルの強度をもつソフトウェアＤＲＭを構築するために暗号鍵 Ktrb で暗号化されたままの Kd（データ暗号化鍵）をハードディスクやフラッシュメモリなどに保存し管理する必要がある場合もあることから、ＴＲＢの暗号鍵 Ktrb は、ＧＴ１０の電源切断後も耐攻撃領域内でＦｅＲＡＭ (Ferroelectric Random Access Memory) 等の不揮発性メモリなどに維持し続

けてもよい。

暗号化キーテーブルとＴＲＢの内容の同期方式にはつぎのような方式が考えられるが、本発明がそのいずれかに限定されるものではない。

（ａ）暗号化キーテーブルへのＴＲＢの内容の書き出しと暗号化キーテーブルからＴＲＢへの再取り込みは特定のレジスタへのＲＷ命令などを介して行う。

（ｂ）暗号化キーテーブルの先頭アドレスと行数を設定する（特別なインストラクションを実行する）ことにより、必要になった際にＧＴ１０が自動的にテーブルとＴＲＢの同期をとる。また、ユーザが必要な時に同期がとれるようにフラッシュインストラクションをＧＴ１０に実装する。

一方、ＴＬＢの内容は、一般のＣＰＵと同様にＧＴ１０でも、外部記憶装置にページテーブルとして記録し、ＧＴ内のＴＬＢの内容とページテーブルの内容とは同期させて管理される。そして、ＧＴ１０は、必要な情報を必要な時にページテーブルからＧＴ１０内に取り込んで利用することとしてもよい。

以下、ＴＬＢとページテーブルの間での内容の受け渡しについて説明する。ＴＬＢの内容をページテーブルに記録させる際には、TLB.tr（ＴＬＢ内の tr フィールドの値）がオンの場合、ＧＴ１０は、TLB.sign（ＴＬＢ内の署名 sign フィールドの値）を生成し、記録させたい内容につける。そして、ページテーブル内の内容をＧＴ内のＴＬＢに取り込む際には、ＧＴ１０は、内容に付されている署名を確認し、署名が正しくなければ、ＴＬＢ署名不正例外（後述）を発生させ、その行の有効性フラグ（TLB.p）と耐攻撃性フラグ（TLB.tr）を無効（オフ）にする。

図９に署名方式の一例を示す。図９に示すように、まず、ＧＴ１０は、TLB.vpn、TLB.rights、TLB.kid 及び TLB.ebim を含むデータにＧＴ１０内の固定の乱数を連結する。続いて、連結されたデータを SHA-1 でハッシュし、ＧＴ１０内の秘密鍵 Ktlb を用いて暗号化する。これにより署名 TLB.sign を生成す

る。署名を付す対象となる内容が 160 ビットに満たない場合、例えば埋め草としての乱数フィールドをその内容に追加する事としてもよい。

以下、図１０を用いて、ＴＲＢ、ＴＬＢ、暗号化キーテーブル及びページテーブルの関係について説明する。

図１０に示すように、ＴＲＢ及びＴＬＢは、ＧＴ１０内に保持される。ＴＲＢには、鍵識別情報 kid、コードブロックを復号する際に用いられる Kc 及びデータブロックを暗号化・復号する際に用いられる Kd 等が記録される。ＴＬＢには、領域識別子 id、鍵識別情報 kid、仮想ページ番号 vpn、物理ページ番号 ppn、アクセス権限 rights 等が格納される。領域識別子 id は、コードブロックやデータブロックが格納されているＲＡＭ１７内のページに対応する。また、ＴＲＢとＴＬＢの内容は、鍵識別情報 kid によって対応付けられている。

ＴＲＢの内容を暗号化キーテーブルに記録する場合、その内容に乱数を追加してから暗号鍵 Ktrb を用いて暗号化する。暗号化キーテーブルの内容をＧＴ１０内に取り込む場合、暗号鍵 Ktrb を用いてその内容を復号する。一方、ＴＬＢの内容をページテーブルに記録する場合、ＧＴ１０は、暗号鍵 Ktlb を用いてその内容に基づく署名 TLB.sign を生成し、その内容に追加する。

続いて、ＧＴ１０内のキャッシュと、ＴＬＢと、ＴＲＢとの同期について説明する。上記のように、ＴＬＢやＴＲＢには、ページに記録された保護コードや保護データへのアクセス制御に関する情報が記録されている。そこで、同じ仮想アドレスをもつ２つのＴＬＢをハードウェアまたはソフトウェアで偽装することにより、保護コード又は保護データにアクセスしようとする攻撃が考え得る。このような攻撃に対処するため、ＧＴ１０において、プロセッサコア１１は、キャッシュとＴＬＢとＴＲＢの内容の同期を取りながら、アクセス制御の判断を実施することとしてもよい。より具体的には、命令キャッシュ１２内の保護コード又はデータキャッシュ１３内の保護データの内容は、ＴＬＢ行の

仮想アドレス(TLB.vpn)だけでなく、そのＴＬＢ行のデジタル署名(TLB.sign)の値と仮想アドレスのペアにリンクすることとしてもよい。この場合、プロセッサコア１１は、このペアが不一致の際には異なるページとしてアクセス制御の判断をおこなう。

このように、GT ライセンスにアクセス許可保護コードのコード復号鍵(Key)を被許諾コード鍵(Authorized Code Key)として埋め込んでおくことで、保護領域にアクセス可能な保護コードを指定することができる。そして、ＴＲＢの Authorized Code Key フィールド(ackey)で指定された Key をもつ仮想ページ上のコードを実行したコードからのみ保護コード及び保護データにアクセス可能とする。ただし、アクセス権がＸまたはＰＷＸ のコードを実行する場合に限り、そのページの ackey の値がすべて０であれば、すべてのコードからの実行を許諾するものとする。

ページＡの ackey として指定された鍵に一致する key の値をページＢが持つとき、ページＢ上の保護コードの実行状態をここではページＡ（上のコード／データ）の「オーナープロセス(Owner Process)」と呼ぶ。

なお、ＧＴにおける実行権（Ｘ）とは、TRB.ackey（ＴＲＢの ackey フィールド）に値を持つ耐攻撃ページ上の命令コードを実行する権利を意味する。この権利はその命令コードのひとつ手前で実行された命令コードに対して与えられるもので、つぎのようなケースがある。

（ａ）ひとつ手前で実行された命令のあとに現在実行すべき命令とが続いている場合。

（ｂ）ひとつ手前で実行された命令が CALL や JUMP などの分岐命令である場合。

特に前の命令と次の命令の仮想ページが異なる場合、ＧＴ１０は、あらためて次に実行することを指定された命令コードが実行を許諾されているかどうか

を確認しなければならない。この確認については後述する。

続いてＴＲＳＳフラグについて説明する。図１１に、ＴＲＳＳフラグを格納するフィールドの構成の一例を示す。ＧＴ１０はその仮想メモリ空間のセグメント指定レジスタに、ＴＲＳＳフラグを持つ。このフラグはコード・セグメント、データ・セグメント、スタック・セグメントのそれぞれに対して存在し、それぞれ耐攻撃コードセグメントセレクタ(TRCSS: Tamper Resistant Code Segment Selector)、耐攻撃データセグメントセレクタ(TRDSS: Tamper Resistant Data Segment Selector)、耐攻撃スタックセグメントセレクタ(TRSSS: Tamper Resistant Stack Segment Selector)と呼ぶ。ＴＲＳＳフラグがオンである場合、ページは耐攻撃空間内に設定され、オフである場合、ページは一般の仮想空間内に設定される。

続いて、ＲＡＣＴについて説明する。図１２に、ＲＡＣＴの各行のフィールド構成の一例を示す。ＧＴ１０によれば、レジスタの封印と解放の機能を実現するために、耐攻撃モジュール内にＲＡＣＴを備える。ＲＡＣＴの各行は、レジスタＩＤrid、封印フラグ seal 及び被許諾コード鍵 ackey 等のフィールドを含む。

レジスタＩＤrid は、レジスタを指定するＩＤである。封印フラグ seal、レジスタが封印中であるか否かを示し、オン（１）である場合レジスタは封印中であり、オフ（０）である場合レジスタは解放されている事を示す。非許諾コード鍵 ackey は、ＴＲＢの ackey と同様であり、レジスタへのアクセスが許可されているプロセスのコードページの鍵 key である。

続いて、ＲＳＳＬレジスタについて説明する。ＲＳＳＬレジスタは、ＧＴに必須の機能ではないが、オプション機能として備えることとしてもよい。ＲＳＳＬレジスタは、各レジスタの封印状態を示す情報を格納する。図１３にＲＳＳＬレジスタのフィールド構成の一例を示す。図１３に示すように、ＲＳＳＬ

レジスタは封印可能なレジスタの数と同じビット長を有し、各ビットは、そのビットに対応するレジスタの封印状態を示す。ビットがオン（１）である場合、当該レジスタが封印されていることを示し、オフ（０）である場合、当該レジスタが解放されていることを示す。例えば、ＲＳＳＬレジスタの第１ビットがレジスタｒ１の封印状態を示すように設定されている場合に、第１ビットがオンであれば、レジスタｒ１は封印されていることを意味する。

続いて、耐攻撃セグメント空間及び耐攻撃領域について説明する。上述のように、ＴＬＢ内の耐攻撃性フラグ tr がオン（１）であるページは、耐攻撃セグメント空間内に含まれる耐攻撃領域である。また、このページに対応するセグメント指定レジスタ内の行において、ＴＲＳＳフラグはオンである。耐攻撃領域内のページには、保護データが格納される。図１４に、一般の仮想セグメント空間と耐攻撃セグメント空間との間でコード実行プロセスからデータにアクセスする際のポリシーを示す。一般仮想セグメント空間は、保護をする必要がないデータ及びコードが記録される空間であり、耐攻撃セグメント空間は保護データ及び保護コードが記録される空間である。

耐攻撃セグメント空間で実行されるコードから一般仮想セグメント空間内のデータにアクセスすることはできるが、一般仮想セグメント空間で実行されるプロセスから耐攻撃セグメント空間内のデータにアクセスはできない。また、ユーザレベル空間とスーパーバイザレベル空間とのアクセス制御関係は、耐攻撃セグメント空間内でもライセンスオーナーが同じであれば、一般仮想空間における場合と同じポリシーが維持される。

また、耐攻撃空間内であっても、ライセンスオーナーが異なる保護コード及び保護データの領域間では、相互にアクセスすることができない。ただし、後述の耐攻撃領域共有機能を用いることで、相互アクセスを可能にすることができる。

上述のアクセスポリシーにより、ＧＴ１０における保護プロセスは、他のプロセスからの影響を受けない。ＧＴ１０において、保護プロセスは自身が生成した耐攻撃ページを持ち、一般にＯＳによって管理される。

図１５に、ＧＴ１０上で保護プロセスが動作する様子を示す。図１５に示すように、ＧＴ１０上では、複数の保護プロセスが動作し、それぞれの保護プロセスは耐攻撃領域内にページを生成する。作業データは、ＧＴ１０内の暗号エンジン１６によって暗号化されてから耐攻撃領域（仮想メモリ領域）に格納される。従って、仮想ハードウェアＴＲＭの範囲は、ＲＡＭ１７やハードディスク等までも伸び縮みし、一定の形を持たないといえる。このため、ＧＴ１０が生成し動作する保護プロセスは「仮想ハードウェアＴＲＭ（Virtual Hardware Tamper Resistant Module :VHTRM)」内で動作しているということができる。故に、ＧＴ１０によれば、ハードウェアＴＲＭ並みの強度を持ちつつも、ハードウェアＴＲＭに伴うリソースの制限という問題を解消することが可能となる。

また、保護プロセスは、世界中のＣＰＵ（ＧＴ１０）空間を自在に移動することも可能である（後述のＧＲＩＤ計算）。このように、どのような攻撃にも耐え、自分の使命を果たして元のＧＴ１０に帰ってくるＶＨＴＲＭを纏った保護プロセスを擬人化して「耐攻撃エージェント(Tamper Resistant Agent)」と呼ぶこともできる。

＜インストラクションセット＞

次に、ＧＴ１０に与えられるインストラクションセットについて説明する。

まず、ＧＴを実装するＣＰＵが、従来のＣＰＵのインストラクションセットに加えて、ＧＴ１０としての機能を実現するために備える基本命令について説明する。基本命令には、以下の（１）から（９）がある。以下、各命令について順に説明する。

（１）ＧＴ証明書取り出し命令：Store GT certificate command

命令形式：STR_GT_CERT (certNum, certAdr)

入力レジスタ：

certNum：ＧＴ１０内でローカルに割り当てられた証明書番号。N 枚の証明書があれば、0 から N-1 までの値が有効となる。

certAdr：証明書の内容を書き込むアドレス

出力（例外処理）：

errorCode：指定した番号の証明書がない場合などに設定される。

動作：certNum で指定された証明書をＧＴ１０内から取り出し、certAdr で指定されたアドレスに書き出す。

動作可能権限：全レベル

（2）アクセス許諾命令：Access authorization command

命令形式：AUTHORIZE (licenseAdr, resultAdr, startVPN, numOfVP, kid)

入力レジスタ：

licenseAdr：ＧＴライセンス（後述）を格納したアドレス

resultAdr：結果を格納するアドレス

startVPN：耐攻撃領域の先頭仮想ページ番号

numOfVP：連続する耐攻撃領域のページ数

kid：ＯＳがＴＬＢとＴＲＢの関係を管理するために割り当てる識別子

出力：

result：［ＧＴライセンス || startVPN || numOfVP］|| 以上の署名］が resultAdr に記録される。

errorCode（例外処理）：認証が失敗した場合などに設定される。

動作：指定レジスタのメモリ上にあるＧＴライセンスを認証し、ＧＴライセンスから取り出した耐攻撃領域のコード又はデータ復号鍵 key (Kc 又は Kd)、kid、ackey を耐攻撃領域内のＴＬＢにリンクするＴＲＢに設定する。さらに、

ＴＬＢに「耐攻撃フラグ」(tr)とアクセス権（PR, X, PRW, PWX）、ebim、kid及び sign を設定する。正しく設定が完了すれば、入力データを連結してそのデジタル署名を付加し、resultAdr で指定されたアドレスに記録する。このとき、入力データの連結をハッシュし、結果を Kgt を用いて暗号化し、その結果をデジタル署名として採用する。認証が失敗した場合やＴＬＢが存在しない場合など、設定が失敗した場合には例外を発生させる。

動作可能権限：スーパーバイザレベル（権限レベル０）のみ

（３）耐攻撃権限設定命令： Set Tamper Resistant Rights command

命令形式：SET_TR_RIGHTS (rights, startVPN, numOfVP)

入力レジスタ：

rights：アクセス権（ACgt.ar（後述）を指定）

startVPN：耐攻撃領域の先頭仮想ページ番号

numOfVP：耐攻撃領域のページ数

出力（例外処理）：

errorCode：入力に対応する行がＴＬＢ又はＴＲＢにない場合、又はすでに設定されているアクセス権よりも指定されたアクセス権の範囲が広い場合などに発生。

動作：startVP 及び NnumOfVP で指定された仮想領域が耐攻撃空間である場合、その領域のＴＬＢに、rights で指定されたアクセス権を設定する。ただし、rights で指定されたアクセス権は、すでにＴＬＢに設定されているアクセス権の範囲でなければならない。

動作可能権限：スーパーバイザレベル（権限レベル０）のみ

（４）耐攻撃例外アドレス設定命令：Set Tamper Resistant Exception command

命令形式：SET_TR_EXCEPTION (startAdr)

入力（イミーディエイトまたは直接アドレッシングのみ）：

startAdr：ＴＲＢに設定する耐攻撃空間内の仮想アドレス。レジスタ解除・復帰例外などのジャンプ先として指定する。なお、封印したいレジスタの使用前に startAdr を秘匿して実行する必要があることから、レジスタは入力パラメタとして利用できない。

出力（例外処理）：

errorCode：staetAdr で指定されたアドレスに対応する行がＴＲＢ内に存在しない（耐攻撃モードではない）。又はそのアドレスへのアクセス権がない。

動作：指定された startAdr を現在実行中のコードページのＴＲＢ.eadr（ＴＲＢ内の eadr フィールド）に設定する。

動作可能権限：全レベル

（５）レジスタ封印命令：Seal Register command

命令形式：SEAL_REG（registerList）

入力レジスタ：　registerList：封印するレジスタ一覧（フラグリスト）。各レジスタに対応したフラグについて、オン（１）でレジスタを封印することを示し、オフ（０）で封印しないことを示す。

出力（例外処理）：

errorCode：指定したレジスタが存在しない。または、そのレジスタはすでに封印済である。

動作：指定したレジスタには、本命令を出したプロセス（コード実行状態）コードページの暗号鍵 Kc と同じ鍵を持つコードページのプロセス以外からアクセスできなくなる。本命令の実行によって、対応するレジスタの RACT.ackey（RACT の ackey フィールド）に、本命例を実行したコードを暗号化した key の値を記録する。

動作可能権限：全レベル

（6）レジスタ解放命令：Release Register command

命令形式：RELEASE_REG (registerList)

入力レジスタ：

registerList：封印を解放するレジスター一覧

出力（例外処理）：

errorCode：指定したレジスタが存在しない。または、そのレジスタは解放済である。

動作：指定した範囲のレジスタの値をすべて０にクリアし、封印を解除する。

動作可能権限：全レベル

（7）非許諾領域スタックストア命令：Store Stack to Unauthorized Area command

命令形式：STMEA_TO_UA (registerList)

入力：

registerList：スタックに記録されるレジスタの一覧。フラグのリスト。各レジスタに対応したフラグがオン（1）の場合、そのレジスタをスタックに記録することを示し、オフ（0）で記録しないことを示す。

出力（例外処理）：

errorCode：registerList で指定されたレジスタが封印されていない。

動作：スタックポインタが、許諾されていない耐攻撃領域を示す場合でも、指定されたレジスタの RACT.ackey（ＲＡＣＴ内の ackey フィールド）の値がその領域に対応するＴＲＢ.ackey（ＴＲＢ内の ackey フィールド）に一致すれば、指定されたレジスタがスタックポインタの位置に記録される。

動作可能権限：全レベル

（8）非許諾領域スタックロード命令：Load Stack from Unauthorized Area command

命令形式：LDMEA_FROM_UA (registerList)

入力：

registerList：スタックから読み出すレジスター一覧

出力（例外処理）：

errorCode：指定レジスタが封印されている。

動作：スタックポインタが耐攻撃領域を示す場合、指定されたレジスタのRACT.ackey にスタックポインタの領域のＴＲＢ.ackey の値がコピーされ、指定されたレジスタの RACT.seal（ＲＡＣＴ内の seal フィールド）がオン（１）に設定される。さらに、指定されたレジスタにスタックポインタの位置のデータがロードされる。

動作可能権限：全レベル

（９）ライセンス削除命令：Delete License commnad

命令形式：DELETE_LICENSE(kid)

入力：

kid：ＯＳがＴＬＢとＴＲＢの関係を管理するために割り当てた識別子で、AUTHORIZE 命令のパラメタとして用いた値。

出力（例外処理）：

errorCode：kid が誤っている場合、又は削除に失敗した場合等に設定される。

動作：指定された kid のＴＲＢ及びそれにリンクするすべてのＴＬＢが削除され、対応するページテーブルと暗号化キーテーブルの有効性フラグもオフに設定される。また、それらのＴＬＢが示していた仮想領域のキャッシュロックも解除される。

動作可能権限：スーパーバイザレベル（権限０レベル）のみ

上述の（１）から（９）までの基本命令に加えて、性能や機能の向上のため

の拡張命令として、（１０）から（１５）までの命令等が考えられる。以下、順に拡張命令について説明する。

（１０）保護ブロック開始命令：Protected Block Start command

命令形式：PRT_BLOCK_START (encryption_flag)

入力（直接パラメタ）：

encryption_flag：オン（１）ならブロックが暗号化されていることを示す。

動作：

命令データの特定のビット位置が特定パターンにセットされていれば暗号化ブロック開始命令とする。本命令内の特定ビット列（７ビット程度）は暗号化ブロックの長さを指定するために用いられ、値を指定する単位は、キャッシュ可能なインストラクション長の最小値と同じとする。また指定可能な最大値はキャッシュ可能な長さの最大値と同じとする。本命令のその他のバイトには乱数を入れなければならない。本命令データの先頭の境界は命令キャッシュ単位の境界に一致しなければならない。

乱数の長さがセキュリティ上十分でない場合は、この命令を２つ以上連続する。連続した場合、最後の本命令以外はブロック長の値に０が入る。

保護コードブロック開始命令は、ＧＴ１０内で復号された後、同じ長さのNop命令となる。

TLB.ebim の第２ビットがオンの場合、ブロックに含まれるハッシュまたは署名（平文の場合）をチェックしなければならない。ハッシュまたは署名の位置は、例えばブロックの末尾とする。ブロックがハッシュや署名を含まない場合は、ハッシュ非実装例外を発生させる。

なお、この命令は、第２実施形態（後述）に対応する。

（１１）ＴＲＢ暗号鍵リフレッシュ命令：Refresh TRB Key command

命令形式：REFRESH_TRB_KEY

入出力レジスタ：なし

動作：ＴＲＢを外部に記録する際にはかならず用いる対称暗号法の暗号鍵 Ktrb とページテーブルの署名確認に用いる暗号鍵 Ktlb を新たな乱数値に置き換える。

この命令は、Ktlb や Ktrb が漏洩する可能性がある場合などに使用する。ただし、ＴＲＢを用いてライセンス情報などを暗号化して管理していた場合、この命令後は以前の暗号化情報を復号できなくなるので、注意が必要。このような場合には例えば、リフレッシュの前に別の鍵で暗号化して退避しておくなどの処理が必要になる。利用例の詳細は後述。

（１２）保護領域のキャッシュを優先する命令：

保護データの暗号・復号速度を重視する場合に利用する。

（１３）プロセス間で漏洩しない熱乱数生成命令：

（１４）内蔵暗号・復号コマンド

ＧＴ１０を実現のためにＣＰＵ内部に実装した対称鍵暗号法や公開鍵暗号法の暗号化・復号命令をインストラクションセットとしてソフトウェア開発者に見せることにより、ハードウェアリソース利用効率が高まる。

（１５）定期割り込み間隔設定コマンド

ＧＴ内蔵水晶発信機が発生する内部割込みの間隔を設定する。外部の信頼できる時計と安全に同期するセキュアクロックを耐攻撃プロセスとして実現する場合に用いることができる。

＜例外処理＞

ＧＴ１０を実現するＣＰＵは、従来の例外処理に加え、次のような例外処理を実装する。例外処理には、大きく分けて、耐攻撃ページアクセス権関連例外と、耐攻撃領域復帰関連例外と、レジスタ封印関連例外と、ハッシュ関連例外

等とがある。以下、順に各例外処理について説明する。

・耐攻撃耐攻撃ページアクセス権関連例外

（1）証明書指定誤り

この例外処理は指定した番号あるいは識別情報の GT 証明書（Cgt）がない場合に発生する。

（2）ＧＴライセンス認証フォールト

この例外処理はＧＴライセンスの認証に失敗した場合に発生する。

（3）耐攻撃空間ページフォールト

ＴＬＢ内の耐攻撃性フラグ tr がオフ（耐攻撃モードではない）または、そのページに対応する行がＴＲＢ内に無いのに耐攻撃領域へのアクセス命令を実行しようとした。

（4）ＴＬＢ署名不一致例外

TLB.sign（ＴＬＢ内の sign フィールド）の値が、ブロックに含まれる署名と一致しない。本例外が発生した場合、ＧＴ１０は自動的に当該ＴＬＢのページテーブルからのロードを停止する。さらに、ＴＬＢ内の対応する行の有効性フラグ（TLB.p）及び耐攻撃性フラグ（TLB.tr）をオフにする。

（5）アクセス権不正拡大例外

AUTHORIZE 命令で許可されたアクセス権よりも指定されたアクセス権の範囲が広い。本例外が発生した場合、ＧＴ１０は自動的にアクセス権設定命令を拒否する。

（6）暗号化キーテーブルアクセスフォールト

この例外は、ＴＲＢ中に存在しない kid その他の検索キーを指定した場合、又は TRB のページアウトに失敗した場合に発生する。

（7）ライセンス削除フォールト

この例外は、ライセンスの削除に失敗した場合に発生する。

・耐攻撃領域復帰関連例外

（８）耐攻撃コード復帰例外

:RETURN_TO_TAMPER_RESISTANT_CODE_EXCEPTION

call、jump 又は return によって、一般の仮想領域又は耐攻撃領域からコード復号鍵 Kc が異なる耐攻撃領域に実行中アドレスが移動した際には、必ず耐攻撃コード復帰例外が発生する。この例外の処理において、移動先耐攻撃領域の例外アドレスである TRB.eadr（ＴＲＢ内の eadr フィールド）の内容が確認され、そのフィールドに例外アドレスが存在する場合、プロセッサコアは、その例外アドレスにアクセスする。例外アドレスが存在しない場合には、当該ＴＲＢ行が削除され、耐攻撃コード復帰例外アドレス不正例外が発生する。

（９）耐攻撃コード復帰例外アドレス不正例外

この例外は、耐攻撃コード復帰例外アドレスで指定された空間が、耐攻撃空間に存在しない場合に発生する。保護プロセスへの復帰時等に本例外が発生した際には、そのアドレスに対応するＴＲＢ内の行の有効性フラグ(p)をオフにし、ＯＳなどによって指定されたアドレスへ強制的にジャンプする。

・レジスタ封印関連例外

（１０）レジスタ封印済例外

この例外は、指定されたレジスタが既に同じ Kc を持つコードにより封印済みである場合に発生する。

（１１）レジスタ解放済例外

この例外は、指定レジスタが封印されていない場合に発生する。

（１２）封印レジスタアクセスフォールト

:SEALED_REGISTER_ACCESS_FAULT_EXCEPTION

この例外は、指定されたレジスタが既に Kc が異なる他のコードによって封印済みである場合に発生する。

・ハッシュ関連例外

（１３）ハッシュ非実装例外

この例外は、ＧＴ１０がハッシュ生成機能やチェック機能を実装していない場合に発生する。

（１４）ハッシュ不一致例外

この例外は、ＧＴ１０が、保護コード又は保護データをキャッシュする際にハッシュ値の不一致が生じた場合に発生する。

＜動作＞

以下、ＧＴ１０によって行われる処理について説明する。

＜ＧＴの認証＞

まず、ＧＴ１０の認証手順について説明する。なお、以下の説明において、ＰＫＩの内容の理解を前提としている。

図１６に、ＧＴの認証を説明する図を示す。図１６に示すように、ＧＴの認証には、保護アプリケーション５０、ＤＲＭ３０及び４０、暗号モジュール（ＰＫＩライブラリ２０に含まれる）の各ソフトウェアパッケージを作成する各開発メーカー、ＧＴ１０を作成する開発メーカー、ＰＫＩライブラリモジュール用認証局 CApki、ＤＲＭ用認証局 CAdrm、ＤＲＭを利用するアプリケーション向け認証局 CAap 及びＧＴを認証する認証局 CAgt が介在する。図１６において、細い矢印は、データの流れを示し、太い矢印は、ソフトウェアパッケージへのライセンス又は証明書の埋め込みを示す。また、図１６中の括弧中の数字は処理の順番を示す。また、各ＴＲＭソフトウェアは、各ＴＲＭソフトウェア開発メーカーが秘密裏に生成した対称鍵暗号方式の暗号鍵で暗号化されているものとする。

ＧＴライセンスの生成と利用は次のような各システム運用とプログラムソフトウェア実行許諾手順によって実現される。

（1）ＧＴメーカーは製品化するＧＴ１０の個別公開鍵 KPgt をＧＴ専用の認証局 CAgt に渡す。さらに、ＧＴメーカーは、ＧＴ１０内部には秘密鍵 Kgt を埋め込む。

（2）ＧＴ専用の認証局 CAgt は、ＧＴ１０の公開鍵証明書 Cgt をＧＴメーカーに発行する。なお、ＧＴ１０の公開鍵証明書 Cgt は、クラス秘密鍵 Kcgt で署名され、クラス公開鍵 KPcgt は、ＧＴ専用の認証局 CAgt のルート秘密鍵 Kagt で署名された証明書の形で公開されていることとしてもよい。また、公開鍵証明書 Cgt において個別鍵とクラス鍵の双方を用いるのではなく、いずれか一方を用いることとしてもよい。クラス鍵がない場合、公開鍵証明書 Cgt はルート秘密鍵 Kagt で直接に署名されることとしてもよい。

（3）ＧＴメーカーは、公開鍵証明書 Cgt をコピーし、ＴＲＭソフトウェア開発メーカーに配布する。

（4）各ＴＲＭソフトウェアの開発メーカーは、開発したソフトウェア実行コードを鍵 Kc を用いて暗号化することにより、保護コードファイルを作成する。

（5）各ＴＲＭソフトウェアの開発メーカーは、公開証明書 Cgt から取り出したＧＴ１０の公開鍵 KPgt を用いて鍵 Kc を暗号化し、（移動不可能な）ＧＴライセンス Lxx を生成する。ＧＴライセンスの構造については後述する。

（6）各ＴＲＭソフトウェアメーカーは、ソフトウェアが実行される前にＯＳのイニシエーターがＧＴライセンス Lxx をＧＴ１０に投入するように、ＧＴライセンス Lxx をそのソフトウェアパッケージに埋め込む。

（7）各ソフトウェアの実行直前にそのＧＴライセンスがＧＴ１０に投入される。ＧＴ１０は、公開鍵のペアにあたる秘密鍵 Kgt を用いて、ＧＴライセンスを復号する。つまり、ＧＴ１０内で各ソフトウェア開発メーカーによるＧＴ１０の認証が実行される。これにより、ＧＴ１０上でそのソフトウェアを実

行することが許諾される。ＧＴで保護されるソフトウェアをフリーソフトウェア等にして流通させたい場合には、ＧＴライセンスを生成するためにクラス公開鍵 KPcgt を用いる事としてもよい。また、ＧＴで保護されるソフトウェアを特定のＧＴ１０以外で使用できないようにしたい場合は、ＧＴライセンスを生成するために個別公開鍵 KPgt を用いる事としてもよい。

（８）ＧＴ１０は、ＧＴライセンスから取り出した鍵 Kc とアクセス権（Access rights）をＧＴ１０内のＴＲＢ及びＴＬＢに設定する。ユーザは、たとえカーネルモード（スーパーバイザレベル／リング０）でも、ＴＲＢに設定された Kc を参照することはできない。

このように、ＴＲＭソフトウェアパッケージには、ＧＴ１０に実行許諾を与えるためのＧＴライセンスが埋め込まれ、暗号化されたプログラムコードが実行される前にそのＧＴライセンスは、ＧＴ１０に投入される。ＧＴ１０は、個別秘密鍵 Kgt を用いてＧＴライセンスを復号し、そのＧＴライセンスからプログラムコードの復号鍵 Kc を取り出し、内部のＴＬＢにリンクしたＴＲＢに、保護コードごとにその復号鍵 Kc を維持する。ＧＴ１０において、暗号化されたコードは復号鍵 Kc を用いて復号されながら実行される。

ＰＫＩライブラリモジュール用認証局 CApki、ＤＲＭ用認証局 CAdrm とＤＲＭを利用するアプリケーション向け認証局 CAap から構成されるＴＲＭソフトウェアパッケージ向け認証局は、各パッケージのソフトウェアプロセスが保護データを交換する際にデータ転送先パッケージを認証するために利用する証明書を発行する。また、これらのうちＤＲＭ用認証局はライセンス転送先ＤＲＭの認証に用いられることしてもよい。

ここで示した４種類の認証局はすべて異なる運用にすることでコンテンツ（情報）流通ビジネスのリスクが軽減することが可能となるが、そうした運用は必須ではない。なお、ＰＫＩライブラリ２０、デコーダＤＲＭ３０及びメデ

ィアＤＲＭ４０を含めて１つのＯＳとして評価し、このＯＳ製品全体に対してＧＴライセンスを生成し、また公開鍵証明書を発行することとしてもよい。

また、ＧＴライセンスを保護実行コードと同じファイルに挿入することとしてもよいが、パッケージ超流通の便を考慮すると、暗号化された実行コードと別のファイルとして管理することを推奨する。

次に、ＧＴライセンスの構造について説明する。

ライセンス生成側はライセンス許諾先ＧＴ（耐攻撃容器）の個別公開鍵 KPgt の証明書 Cgt を取得し、これを用いてＧＴライセンスを生成する。証明書 Cgt は X.509 準拠の形式としてもよい。個別公開鍵 KPgt の証明書 Cgt はクラス鍵 KPcgt で、また、クラス鍵 KPcgt の証明書 Ccgt は認証局 CAgt のルート公開鍵 KPagt で署名チェックした後、利用される。ライセンス生成側が生成する GT ライセンスの形式は以下のとおりである。

```
LicenseID || ACgt.ar ||
    E (KPgt, Ks ) || E ( Ks, Key || LicenseID || ACgt || Is ) ||
    CgtID || CgtIDackey || else
```

ここで、各フィールドの意味は次のとおりである。

E (K, D)：データ D を鍵 K で暗号化した結果

A || B：データ連結。データ A とデータ B が連結していることを示す。

Ks：セッション鍵（乱数）。対称鍵（共通鍵）暗号法の鍵。

Key：コード復号鍵／データ暗号化・復号鍵。Kc/Kd。

LicenseID：ライセンスシリアル番号。ライセンス生成側がライセンスごとにユニークな番号を生成する。上位にライセンス生成者の ID を入れ、中位にコンテンツの ID を入れることとしてもよい。

ACgt：ＧＴアクセス条件。ＧＴ内で強制可能なコード／データの利用条件を示し、次のフィールドを持つ。

ebim：暗号化ブロック識別方式。図５に示す TLB.ebim フィールドにコピーされる。各値の意味はＴＬＢのフィールドとして説明済みである。

aef：被許諾コード鍵暗号化フラグ(ackey encryption flag)。on(1)なら被許諾コード鍵 ackey が個別公開鍵 KPgt を用いて暗号化されていることを示し、off(0)なら ackey の値がそのまま ACgt.ackey（ACgt の ackey フィールド）に入っていることを示す。

ackey：被許諾コード鍵。暗号鍵 Kc 又は Kd で復号されたオブジェクトにアクセスが可能なプロセスのコードが記録されている保護ページの暗号鍵 key または暗号鍵 key を個別公開鍵 KPgt で暗号化した値。個別公開鍵 KPgt で暗号鍵 key を暗号化する場合で、GT が複数の KPgt を持つ場合には、さらにどの KPgt を用いているかを識別できる情報を付加する。なお、そのプロセスへのアクセス権の種類は、ar で指定される。

eadr:例外処理アドレス。このフィールドはオプションである。key が異なるページからこのＧＴライセンスの内容が設定されたページに復帰する直前に発生する例外処理コードの先頭アドレスである。

ar：基本アクセス権。次のような値をとる。詳細は図１７に示す。

(a) PR：ackey で指定されたプロセスからの読み出しのみ可能

(b) X：ackey で指定されたプロセスからの読み出しと実行が可能

(c) PRW：ackey で指定されたプロセスからの読み出しと書き込みが可能

(d) PWX：ackey で指定されたプロセスからの読み書きと実行が可能

uo：外部出力禁止フラグ。on(1)ならこのライセンスのキー情報を格納したＴＲＢ行は暗号化キーテーブルに出力(ページアウト）されない。このフラグのデフォルトはオフ(0)であり、外部に出力可能であることを示す。

cl:キャッシュロックフラグ。オプション機能である。このフラグがオンである場合、このＧＴライセンスで保護を指定された耐攻撃領域のデータはキャッシュの外に出ない。但し、ar が書き込み権を含まない（PR 又は X である）場合には、本フラグは無効となる。このフラグのデフォルト値は、オフであり、外部に出力可能である事を示す。

df：デバッグモードフラグ(Debug mode Flag)。df が on なら、ar の指定に応じた動作を示す。ＴＬＢについての説明参照。なお、ＧＴライセンス内の ACgt の df をオンにする事により、保護コードをデバッグモードで実行する事ができる。また、超流通など形態で保護コードファイルを販売する場合には、df をオフにして販売することとしてもよい。

CgtID：KPgt の X.509 証明書の識別子値。issuer と serialNumber を連結した値。

CgtIDackey：オプション。ackey を暗号化した KPgt の X.509 証明書の識別子値。issuer と serialNumber を連結した値。ACgt.aef が off の場合には省略。また、CgtID と同値の場合にも省略可能。

Is：その他の情報

図１７に、ＧＴライセンスに設定可能なアクセス権と被許諾権限の表を示す。耐攻撃性フラグ耐攻撃モードの際、プロセス（コード実行状態）からみた保護データ又は保護コード領域のアクセス権は、図１７の中から選択され、ＴＬＢ

に設定される。なお、図１７における「指定コード」とは、当該ＴＲＢ行内のKey フィールド又は Ackey フィールドで指定された鍵で復号可能なコードをいう。

また、図１７において、「実行のみ可能」な権限がないが、ＦＲなど、「実行のみ可能」な権限と「読み出しと実行が可能」な権限の両方を指定できるＣＰＵもある。その場合の耐攻撃権限は、それぞれ、ＰＸ及びＰＲＸと表現する事ができる。同様に、「書き込みが可能」な権限についてもＰＷＸ及びＰＲＷＸの両方が指定できることとしても良い。

ＧＴライセンスは、移動不可を条件としているため、移動機能やＣＲＬ（Certificates Revocation List：証明書失効リスト）及びＬＲＬ（License Revocation List：ライセンス失効リスト）をもたない。ＣＲＬやＬＲＬを制御する必要がないため、ＧＴのＣＰＵへの実装が容易になる。ＤＲＭのＣＲＬ制御及びＬＲＬ制御はＯＳ又はアプリケーションにて行われる。これにより、高い拡張性を維持する事が可能となる。

また、上述のように、ソフトウェアＴＲＭをフリーソフトウェア等にしたい場合、ＧＴライセンスの生成にはクラス公開鍵を用いることとしても良い。また、逆に、特定のＧＴのみにソフトウェアを利用させたい場合には、ＧＴ個別鍵に対応する公開鍵を用いることとしても良い。

以下、図１８を用いて、上述のＧＴ認証処理における（８）の手順について詳しく説明する。ＧＴ認証処理において、ＧＴ１０は、アクセス許諾命令（AHTHORIZE 命令）を実行することにより、ＧＴライセンスに基づいてＴＬＢ及びＴＲＢに鍵Key やアクセス権等を設定する。

そのために、まず、ＧＴ１０は、ＧＴの公開鍵のペアにあたる秘密鍵 Kgt を用いてＧＴライセンスを復号する（Ｓ１）。続いて、ＧＴ１０は、正常にＧＴライセンスを復号できたか否か判定し（Ｓ２）、正常に復号できた場合（Ｓ

２:正常）、ＧＴ１０は、指定仮想領域のＴＬＢを検索する（Ｓ３）。正常に復号できなかった場合(Ｓ２：異常）、ＧＴ１０は、ＧＴライセンス認証フォールトを発生し（Ｓ１１）、Ｓ１６に進む。Ｓ１６において、ＧＴ１０は、エラー内容を resultAdr に設定し、メインフローに復帰する。

Ｓ３においてＴＬＢを検索した結果、指定仮想領域が得られた場合（Ｓ４：正常）、ＧＴ１０はＴＲＢに空きがあるか否か判定する（Ｓ５）。Ｓ３の検索の結果、指定仮想領域が得られなかった場合（Ｓ４：なし）、ＧＴ１０は、メモリが割り当てられていない旨を示す未割り当て例外を発生し（Ｓ１２）、Ｓ１６に進む。

Ｓ５においてＴＲＢの空きがあった場合（Ｓ５：あり）、ＧＴ１０は、GT ライセンスに基づいて、ＴＲＢ内の uo、cl、kid、key、ackey の各フィールドに値を設定する（Ｓ６）。Ｓ５において、ＴＲＢの空きが無かった場合（Ｓ５：なし）、ＧＴ１０は、ＴＲＢ内の一部の行を暗号化キーテーブルに出力（ページアウト）する（Ｓ１３）。ＧＴ１０は、ページアウトに成功した場合（Ｓ１４：成功）Ｓ６に進み、ページアウトに失敗した場合（Ｓ１４：失敗）、暗号化キーテーブルアクセスフォールトを発生させた後（Ｓ１５）、Ｓ１６に進む。

Ｓ６の後、ＧＴ１０は、ＴＬＢに格納する署名 sign を算出し（Ｓ７）、ar、tr、df、kid、ebim 及び sign をＴＬＢに設定する（Ｓ８）。続いて、ＧＴ１０は、設定結果の署名を生成し（Ｓ９）、設定結果とＳ９で生成した署名とを resultAdr に設定し（Ｓ１０）、メインフローに復帰する。

＜保護コードファイルの構造＞

以下、保護コードファイルの構造について説明する。暗号化がほどこされた保護コードファイル(実行形式）は、保護コードブロックおよび平文コードブロックの繰り返し構造の先頭にヘッダを付加した構造を持つ。また、保護コードファイルを超流通に利用可能なファイルとする場合、そのファイルのヘッダ

には、さらに次のような情報が入る必要がある。

・コンテンツＩＤ：保護コードの暗号を解くためのコンテンツ復号鍵を持つＧＴライセンスとのリンク情報として必要である。

・コンテンツ暗号方式：保護コードの暗号化方式を特定する値として必要である。この値は復号鍵の長さを含むこととしてもよい。

図１９に、超流通ファイル形式として用いる場合の保護コード実行形式の一例を示す。図１９において、ＳＣＤＦ（超流通コンテンツ形式：Super Content Distribution Format）ファイルに、保護コード実行形式の内容が、ＳＣＤＦの要素として格納されている。

＜保護コードのロード及び実行並びに保護データの退避と維持＞

以下、図２０を用いて、保護コードのロードと実行手順、及び保護データの退避と維持について説明する。

上述のように、暗号化が施された保護コードファイルは、ヘッダ、保護コードブロック及び平文コードブロックで構成される。保護コードファイルは実行の際にＲＡＭ１７にロードされ、ＧＴ１０（ＣＰＵ）内での予測に従いブロックごとに命令キャッシュ１２にコピーされる。このうち、ヒットした命令のみがプロセッサコア１１で解釈され、実行される。図２０に示すように、コードブロックのうち、保護コードブロックは、復号キャッシュラインで復号されてから命令キャッシュ１２に記録される。

図２０では、命令キャッシュ１２とＲＡＭ１７の間に、暗号ブロックを復号して命令キャッシュ１２に書き込む復号キャッシュライン及び、平文ブロックを命令キャッシュ１２に書き込む平文キャッシュラインの２種のキャッシュラインが備えられている。このように複数のキャッシュラインを備えることにより、処理を高速化することが可能である。

また、図２０に示したように、保護コードの実行によって保護データをＲＡ

Ｍ１７に出力する際には、あらかじめ仮想メモリとして設定された耐攻撃領域に出力する。耐攻撃領域には、保護データを暗号化および復号するための対称鍵暗号法の鍵 Kd が関連付けられ、その鍵 Kd は秘密裏にＧＴ１０内に維持されている。鍵 Kd は、コード復号鍵 Kc ごとに異なる乱数として割り当てられる。保護データは一旦ＧＴ１０（ＣＰＵ）内部のデータキャッシュ１３に記録されたのち、暗号化されてからＲＡＭ１７に出力される。また、ＲＡＭ１７上の保護データは、ＧＴ１０（ＣＰＵ）内部で復号されてデータキャッシュ１３に記録され、そのうちヒットしたものがプロセッサコア１１で利用される。

図２０では、データキャッシュ１３とＲＡＭ１７の間に、暗号ブロックを復号して命令キャッシュ１２に書き込む復号キャッシュライン、データキャッシュ１３の内容を暗号化してＲＡＭ１７内の耐攻撃領域に書き込む暗号キャッシュライン、及び、平文ブロックを命令キャッシュ１２に書き込む平文キャッシュラインの３種のキャッシュラインが備えられている。この構成によっても処理を高速化することが可能である。

なお、暗号化された保護データをＲＡＭ１７からストレージ１８内に退避させることが可能である。つまり、耐攻撃領域はＲＡＭ１７内のみならず、バス１８を介して接続されたストレージ１９内にまでも拡張することが可能である。

＜ページアクセス制御＞

以上のようにして、ＧＴ１０は、ＴＲＭプログラムコードを実行する許諾を得て、その保護コードファイルを構成するコードブロックをＲＡＭ１７にロードすると、保護コードブロックを復号して実行する。以下、保護コードを実行する際の保護コードを記録するページ（命令ページ又はコードページ）へのアクセス制御について図２１を用いて説明する。図２１において、矢印は、データの流れを示し、括弧内の数字は、処理の順番を示し、以下の説明における手順の番号に対応する。

保護コードを記録するページへのアクセス制御は次のように行われる。

（１）命令MMU１４が、予測命令ポインタ(ip: instruction pointer)に記憶されている仮想アドレスを読み出す。

（２）命令MMU１４は、仮想アドレスに対応する物理ページ番号 ppn と耐攻撃フラグ tr と耐攻撃モードのアクセス権 ar をＴＬＢから読み取る。

（３）耐攻撃フラグが on の場合、命令MMU１４は、ＴＲＢからコード復号鍵(Kc)を取り出す。

（４）命令MMU１４は、暗号エンジン１６にKc を設定する。

（５）命令MMU１４は、キャッシュすべき保護コードブロックのアドレスを命令キャッシュ１２とＲＡＭ１７に指定する。

（６）ＲＡＭ１７から保護コードブロックを暗号エンジン１６に投入する。

（７）暗号エンジン１６で復号された保護コードブロックは、命令キャッシュ１２に記録される。

（８）命令ポインタが記憶する仮想アドレスが予測命令ポインタに一致した場合（つまりキャッシュヒットした場合）、プロセッサコア１１は ip 用レジスタからＴＲＳＳフラグを読み出し、次の手順に進む。一方、キャッシュヒットしなかった場合、上述の(２)の手順に戻る。

（９）プロセッサコア１１は、ip用レジスタから仮想アドレスを読み出す。

（１０）プロセッサコア１１は、命令MMU１４からアクセス権 ar を取り出し、アクセス権を確認する。

（１１）プロセッサコア１１は、仮想アドレスの内容が記録された命令キャッシュ１２内から保護コードブロックを読み出して実行する。

なお、手順（１０）において、実行する命令を記録する命令ページが切り替わった時には、プロセッサコア１１は、手順（１１）で保護コードブロックの読み出しを行う前に、以下を確認する。

（ａ）次に実行する保護コードブロックが記録されているページの暗号鍵(TRB.key)または被許諾コード鍵（TRB.ackey)が、直前に実行したコードブロックが記録されているページの暗号鍵(TRB.key)と一致すること

（ｂ）次に実行する保護コードブロックについての TLB.ar に記録されたアクセス条件と、次に実行するアクセスとが合致すること

（ａ）及び（ｂ）の少なくともいずれかが不一致の場合には、プロセッサコア１１は命令の実行を停止し、アクセス権例外を発生させる。

次に、保護データブロックが記録されているページへのアクセス制御について説明する。保護データブロックへのアクセス制御は、ＧＴライセンスの内容が TRB 及び TLB に設定されることによって強制される。保護データブロックへのアクセスは TRB 及び TLB に維持されている被許諾コード鍵 ackey、暗号・復号鍵 Kd、アクセス権 ar に従って制御される。

上述の保護コードブロックの場合のアクセス制御手順と、実行中の保護プロセスによる保護データブロックへのアクセス制御手順は、同様である。また、上記の手順（１０）において、実行する命令を記録するページが切り替わったとき、及びアクセスされる保護データを記録するページが切り替わったときには、保護データへのアクセスを実行する前に、プロセッサコア１１は、次の点を確認する。

（ａ）アクセスされる保護データを記録するページの暗号鍵(TRB.key)または被許諾コード鍵（TRB.ackey)が実行コードを記録するページの暗号鍵(TRB.key)と一致すること

（ｂ）アクセスされる保護データを記録するページのアクセス権（TLB.ar)と、次に実行するアクセスとが合致すること

（ａ）及び（ｂ）の少なくともいずれかが不一致の場合には、プロセッサコア１１は、保護データへのアクセスを中止し、アクセス権例外を発生させる。

以下、図２２を用いてプロセッサコア１１によって行われる上述の保護コード及び保護データへのアクセス制御について、より詳しく説明する。

まず、プロセッサコア１１は、例外／割り込みイベントの発生を待つ（Ｓ２１）。例外／割り込みイベントが発生すると、プロセッサコア１１は、例外・割り込み処理を行い（Ｓ３５）、Ｓ２１に戻る。なお、例外・割り込み処理について詳しくは図２３に示す。図２３に示される各種例外処理については、上記＜例外処理＞において詳しく説明したため、説明を省略する。

例外／割り込みイベントが発生しない場合（Ｓ２１：なし）、プロセッサコア１１は、命令ページ切り替えが発生したか否か判定する（Ｓ２２）。

命令ページ切り替えが発生した場合（Ｓ２２：あり）、プロセッサコア１１は、命令ページへのアクセス権の有無を判断する処理を行う（Ｓ２３）。命令ページ切り替えが発生していない場合（Ｓ２２：なし）、Ｓ２８に進む。

Ｓ２３の判断処理において、プロセッサコア１１は、次に実行する保護コードブロックが記録されているページの暗号鍵(TRB.key)または被許諾コード鍵(TRB.ackey)が、直前に実行したコードブロックが記録されているページの暗号鍵(TRB.key)と一致し、且つ、次に実行する保護コードブロックについてのTLB.ar に記録されたアクセス条件によって次に実行するアクセスが許可されているか否か判定する。上記２つの条件を満たす場合、プロセッサコア１１は、次に実行するコードには、切り替わった命令ページへのアクセス権があると判断し（Ｓ２４：あり）、Ｓ２５に進む。そうでない場合（Ｓ２４：なし）、プロセッサコア１１は、ページアクセスフォールト例外をキューイングし（Ｓ２７）、Ｓ２１に戻る。

Ｓ２５において、プロセッサコア１１は、次に実行する保護コードブロックが記録されているページの暗号鍵(TRB.key)が、直前に実行したコードブロックが記録されているページの暗号鍵(TRB.key)と一致するか否か判定し、さら

に、プロセッサコア１１は、そのページに対応するＴＲＢ内の行の eadr フィールドに既に値が設定されているか否か判定する。TRB.key が一致せず、且つ eadr フィールドに既に値が設定されている場合（Ｓ２５：ＹＥＳ）、プロセッサコア１１は、耐攻撃コード復帰例外をキューイングし（Ｓ２６）、Ｓ２１に戻る。TRB.key が一致するか、又は eadr フィールドに値が設定されていない場合（Ｓ２５：ＮＯ）、Ｓ２８に進む。

Ｓ２８において、プロセッサコア１１は、命令ページから命令を読み出して、その命令を解析する。さらに、プロセッサコア１１は、現在実行されているコードが、命令の中で指定されたレジスタへのアクセス権を有するか否か確認する（Ｓ２９）。そのコードが指定されたレジスタへのアクセス権を有する場合（Ｓ２９：あり）、Ｓ３０に進む。そのコードが指定されたレジスタへのアクセス権を有しない場合（Ｓ２９：なし）、プロセッサコア１１は、封印レジスタアクセスフォールト例外をキューイングし（Ｓ３４）、Ｓ２１に戻る。

Ｓ３０において、プロセッサコア１１は、レジスタで示されたデータページが、前のデータページと切り替わっているか否か判定する。切り替わっていない場合（Ｓ３０：なし）、命令を実行し（Ｓ３３）、Ｓ２１にもどる。なお、命令実行処理について詳しくは図２４に示す。図２４に示される命令については、上記＜インストラクションセット＞において詳しく説明したため、説明を省略する。

データページが切り替わっていると判定した場合（Ｓ３０：あり）、プロセッサコア１１は、そのデータページへのアクセス権の有無を判断する処理を行う（Ｓ３１）。Ｓ３１の処理は、Ｓ２３で説明した保護コードの場合と同様であるので説明を省略する。Ｓ３１の判断の結果、プロセッサコア１１は、次に実行されるコードには、切り替わったデータページへのアクセス権があると判断する場合（Ｓ３２：あり）、Ｓ３３に進む。そうでない場合（Ｓ３２：な

し）、Ｓ２７に進む。

＜保護ソフトウェアの起動＞

次に、図２５を用いて、ＧＴ１０上で保護されるソフトウェア、つまり保護コードファイルを起動する際における、ＯＳカーネル６０及びＧＴ１０の動作について説明する。図２５において、細い矢印がデータのリンクを示し、太い矢印がデータの流れを示す。また、括弧内の数字は、処理の順番を示し、以下の説明における手順の番号に対応する。

保護コードファイルを起動する手順は、以下のとおりである。

（１）ＯＳカーネル６０は、ＴＬＢを設定することにより、保護コード及び保護データをロードする領域となる仮想メモリ領域及び物理メモリ領域を確保し、仮想メモリと物理メモリ領域とをリンクさせる。

（２）ＯＳカーネル６０からの AUTHORIZE 命令に基づいてＧＴ１０は、ＴＬＢにリンクするＴＲＢを設定する。

（３）ＯＳカーネル６０からの CALL などのコードモジュール呼び出し命令に基づいて、ＧＴ１０は、ip レジスタを設定する（呼び出し先インストラクションにヒットしない場合、ＧＴ１０は該当するコードブロックを復号し、復号されたコードをキャッシュにコピーする）。

（４）ＧＴ１０は、ip で指定されたアドレスの命令を実行する権利をTLB.ar（ＴＬＢ内の ar フィールド）の内容で確認する。

（５）命令を実行する権利を確認できた場合、ＧＴ１０は、ip で指定されたアドレスの命令（図では STR 命令）を読み出し、デコードし、実行する。

＜保護データへのアクセス＞

次に、図２６を用いて、ＧＴ保護コードを実行するプロセスから保護データにアクセスするメカニズムについて説明する。この例では、保護データ領域は保護コード実行前に確保され、保護コードファイルから初期値もロードされて

いることを前提としている。また、ＯＳ６０からの AUTHORIZE 命令をＧＴ１０が実行する事により、保護データ領域へのアクセス許諾とアクセス権設定は、既に行われているものとする。

以下、例として、命令"STR r0, [r1]"を実行する（レジスタｒ０の値が示す内容をレジスタｒ１の値が示すアドレスに書き込む命令）場合について、保護プロセスが保護データにアクセスする手順と、それに対応するＧＴ１０の動作について説明する。図２６において、矢印や括弧内の数字の意味は、図２５と同様である。

（１）保護コードは、メモリアクセス命令(STR やLDR など)を実行する。

（２）ＧＴ１０は、アクセスするアドレスに対応するＴＬＢ内の行中のアクセス権情報 TLB.ar（ＴＬＢ内の ar フィールド）を確認する。

（３）保護コードは、TLB.vpn（ＴＬＢ内の vpn フィールド）に一致する仮想ページ番号を持つページのデータにアクセスする。（なお、データがキャッシュにない場合、保護コードは、その仮想ページに対応する物理ページからデータを復号してキャッシュにコピーする。）

＜保護プロセスからの保護モジュールの呼び出し＞

次に、保護プロセスから保護モジュールを呼び出す手順について説明する。保護プロセスから他の保護コードモジュール（ＤＬＬ（Dynamic Link Library）など）を起動する場合には、次の２ケースがある。

（ａ）呼び出される保護コードモジュールのコード復号鍵 Kc が、そのモジュールを呼び出す保護プロセス（以下、親プロセスという）のコード復号鍵と同じケース

（ｂ）呼び出される保護コードモジュールのコード復号鍵 Kc が親プロセスのコード復号鍵と異なるケース

（ｂ）の場合に、保護親プロセスから他の保護コードモジュールを呼び出す

際に行う手順は図２７に示すようになる。図２７においても、矢印や括弧内の数字の内容は、図２５と同様である。なお、図２７において、保護コードモジュールはＤＬＬとなっているが、これは一例に過ぎない。

（１）親プロセスは、ＴＬＢの設定などにより、保護コードモジュールをロードするための仮想領域と、その仮想領域に対応する物理領域を確保する。

（２）親プロセスは、アクセス許諾（AUTHORIZE）命令によりＴＬＢにリンクするＴＲＢを生成してコード復号鍵 Kc を設定し、ＴＬＢにアクセス条件を設定する。

（３）親プロセスは、利用している秘密情報用レジスタを、その親プロセスの保護データ領域内のスタック領域に格納する。（必要に応じ、保護データキャッシュ内のスタックデータは暗号化されて外部メモリに記録される。）

（４）親プロセスは、レジスタ解放（RELEASE_REG)命令を実行する。

（５）CALL 命令などにより、親プロセスは、保護コードモジュールを呼び出す。

（６）親プロセスは、レジスタ封印(SEAL_REG)命令を実行する。

（７）呼び出しが返った場合、親プロセスは、退避したスタックをもとのレジスタに復帰させる。

図２８に、上記手順においてＯＳカーネルが行う処理のフローチャートを示す。

図２７の手順の（１）において親プロセスが、保護コードモジュールをロードするための仮想領域と、その仮想領域に対応する物理領域を確保することを要求すると、耐攻撃領域取得システムコールが呼び出される。この要求の際、親プロセスは、必要となる領域のサイズと、ＧＴライセンスをＯＳカーネル６０に通知する。このシステムコールにおいて、まず、ＯＳカーネル６０は、アプリケーションが利用しているレジスタの一覧を入力パラメタとする非許諾領

域スタックストア（STMEA_TO_UA）命令をプロセッサコア１１に出力する（Ｓ９１）。このＳ９１は、図２７の手順（３）に対応する。

続いて、ＯＳカーネル６０は、アプリケーションが利用しているレジスタの一覧を入力パラメタとするレジスタ解放（RELEASE_REG）命令をプロセッサコア１１に出力する（Ｓ９２）。これにより、指定されたレジスタが解放される。このＳ９３は、図２７の手順（４）に対応する。

ＯＳカーネル６０は、ＯＳが利用しているレジスタの一覧を入力パラメタとするレジスタ封印（SEAL_REG）命令をプロセッサコア１１に出力する（Ｓ９３）。Ｓ９４において、これら命令のパラメタが正しい場合（Ｓ９４：正常）、これらの命令はプロセッサコア１１によって実行され、Ｓ９５に進む。そうでない場合（Ｓ９４：不正）、Ｓ１０３に進む。

Ｓ９５において、ＯＳカーネル６０は、親プロセッサによって指定された領域サイズのエントリをページテーブルに設定する（Ｓ９５）。指定されたサイズの領域を設定するための耐攻撃領域のリソースがあった場合（Ｓ９６：あり）、Ｓ９７に進む。耐攻撃領域のリソースが不足していた場合（Ｓ９６：不足）、Ｓ１０３に進む。

Ｓ９７において、ＯＳカーネル６０は、ＧＴライセンスと、設定したアドレス及びページ領域を入力レジスタに設定し、アクセス許諾（AUTHORIZE）命令をプロセッサコア１１に出力する。このＳ９７は、図２７の手順（２）に対応する。プロセッサコア１１が、正常に命令を実行した場合（Ｓ９９:正常）、Ｓ１００に進む。正常に命令を実行できなかった場合（Ｓ９９：例外発生）、Ｓ１０３に進む。

Ｓ１００において、ＯＳカーネル６０はアクセス許諾（AUTHORIZE）命令の結果を返却データとして設定する。この後、図２７の手順（５）が行われ、保護コードモジュールが呼び出される。続いて、ＯＳカーネル６０は、レジスタ

解放（RELEASE_REG）命令をプロセッサコア１１に出力し、ＯＳが使用しているレジスタを解放させる（Ｓ１０１）。最後に、ＯＳカーネル６０は、被許諾領域スタックロード（LDMEA_FROM_UA）命令をプロセッサコア１１に出力し、スタックされていたデータをレジスタにロードさせ、通常状態に復帰する。このＳ９５は、図２７の手順（７）に対応する。

なお、パラメタの不正、リソースの不足及び例外が発生した場合、Ｓ１０３において、ＯＳカーネル６０はエラー内容を返却データとして設定し、Ｓ１０１に進む。

＜耐攻撃コードの復帰時の例外処理＞

call、jump 又は return 等の命令によって、一般の仮想領域又は耐攻撃領域からコード復号鍵 Kc が異なる耐攻撃領域に実行中アドレスが移動した際には、耐攻撃コード復帰例外が発生する。この例外処理の中で、次の２つの処理を実施する必要がある。

（ａ）レジスタ封印の確認と、レジスタを解放済みの場合の封印の再設定。

（ｂ）耐攻撃セグメントセレクタの値の確認と、必要に応じた再設定。

ＧＴ１０は、保護コードのレジスタ封印の前に、解除・復帰例外の先頭アドレスをＴＲＢに設定する。また、外部からの割り込みによってレジスタが解放されたまま実行を継続することによって、レジスタに記録した秘密情報が漏洩することがないよう、保護コードは解除・復帰例外処理を含む必要がある。この解除・復帰例外処理において、ＧＴ１０は、封印したはずのレジスタが封印されているかを再度確認し、レジスタが封印されていなければ、封印しなおすか、保護コードの実行を中止するかしなければならない。

図２９に、耐攻撃コード復帰例外処理による封印レジスタ不正アクセス防止のためのシーケンス例を示す。図２９において、保護コードが不正なコードによって一時中断された後に復帰する際に、例外が発生した場合を示す。図２９

において、保護コードを中断する前にレジスタ封印（SEAL_REG r1）命令が実行されているが、不正コードの実行中にレジスタ解放（RELEASE_REG r1）命令が実行されたため、復帰時に封印したはずのレジスタが封印されていない。

図２９に示す耐攻撃コード復帰例外のシーケンスの最初の３行は以下のとおりである。

```
TST RSSL.rlss, rlssBit
BNE chk_seg
SEAL_REG r1
```

この３行は、保護コードが利用するレジスタｒ１の封印状態をＲＳＳＬレジスタ内のレジスタｒ１に対応するビット rlss の値を調べる事により確認し、ｒ１が封印されていない場合、ｒ１を封印しなおす処理を示す。

また、不正なコードが割り込み等によってプロセスを奪い、データやスタックのＴＲＳＳフラグをオフ（０）に設定する可能性もある。ＴＲＳＳフラグがオフになっているにもかかわらず保護コードが継続されると、耐攻撃領域ではなく一般仮想領域に保護データを書き込んでしまうことになる。このような事態を防止するために、耐攻撃コード復帰例外処理において、ip(命令ポインタ、ＣＰＵによっては PC（プログラムカウンタ）ともいう）が示す仮想アドレスごとの耐攻撃データ／スタックセグメントセレクタの再設定を行う必要がある。図２９に示す耐攻撃コード復帰例外のシーケンスにおける４行目から６行目は以下のとおりである。

```
ch_seg CMP ip, trSegmentHead
       BMI ret
       ORR trdss, trdss, trBit
```

この３行が、ip が示す仮想アドレスごとの耐攻撃データ／スタックセグメントセレクタの再設定を行う処理を示す。耐攻撃コード復帰例外処理が終了する

と、プロセスは保護コードに復帰する。なお、図２９のシーケンスの最後において、不正なコードによって再度プロセスが中断され、不正なコードが保護コードが利用するレジスタｒ１の内容を一般仮想領域に書き込もうとしている（MOV r1, [unprotectedArea]）。しかし、耐攻撃コード復帰例外処理において、レジスタｒ１は封印しなおされているため、封印レジスタアクセスフォールト（SEALED_REGISTER_ACCESS_FAULT_EXCEPTION）が発生している。

このように、耐攻撃コード復帰例外のシーケンスによって、ＧＴ１０は、復帰時に保護コードの秘密データを守っている。

＜保護コンテキストの切り替えの管理＞

以下、耐攻撃ユーザプロセスからＯＳカーネルプロセスへのコンテキスト切り替えがおこり、全レジスタがスタックに退避されるケースを想定する。図３０に、ＯＳカーネルによる保護コンテキスト切り替え時のシーケンスの一例を示す。

図３０において、アプリケーション１からアプロケーション２に保護コンテキストを切り替えると仮定している。当初、アプリケーション１が利用するレジスタｒ１は、封印されている。その後、保護コンテキストを切り替える際に、ＯＳカーネル６０は、非許諾領域スタックストア（STMEA_TO_UA）命令を実行することにより、レジスタｒ１の値はデータ暗号・復号鍵 Kd で暗号化されてから退避される。さらに、スタックされたレジスタｒ１は、レジスタ解放（RELEASE_REG）命令により０クリアされる。レジスタｒ１が解放されることにより、アプリケーション２のプロセスがレジスタｒ１を利用することが可能になる。

その後、保護コンテキストが元のプロセスにもどる際に、ＯＳカーネル６０は非許諾領域スタックロード（LDMEA_FROM_UA）命令を実行することにより、退避されていたレジスタの値が復号されて、レジスタｒ１に戻される。その後、

レジスタｒ１は再度封印される。コンテキスト切り替えの際に、このような保護コンテキスト管理によって、封印レジスタを含めたレジスタの維持、復帰を保障することはＯＳカーネル６０の処理として位置付けられる。

なお、ＯＳカーネル６０が非保護システムコールを呼び出す際は、SP を一般仮想空間のアドレスに切り替えてから呼び出すなどの工夫が必要である。ＯＳは従来通り、システムコールの戻り値を一般仮想領域に記録すればよい。

一方、ＯＳカーネル６０が保護システムコールを呼び出す際には、戻り値を格納するスタック領域を共有するためのＧＴライセンスをパラメタなどとして渡す。ＯＳは受け取ったＧＴライセンスをパラメタとして AUTHORIZE コマンドを実行することで、耐攻撃スタック領域をアプリケーションプロセスと共有することができる。

＜保護データ領域の安全な共有＞

上記において、耐攻撃空間内のプロセス同士であっても、ライセンスオーナーが異なるコード及びデータの領域間では、相互にアクセスすることができないと説明した。しかし、以下の耐攻撃領域共有機能を用いることで、ＧＴ１０で保護され、互いに異なるコード復号鍵 Kc を持つソフトウェアパッケージ、ライブラリ、オブジェクト及びプロセス等の保護コードモジュール間で、保護データを安全に交換できるようにすることができる。

図３１に保護データ領域の共有の一例を示す。図３１は、ＧＴ１０上でプレーヤプロセス及びデコーダＤＲＭプロセスが動作している場合を説明する図である。

デコーダＤＲＭプロセスによってデコードされたデータは、ＤＲＭプロセス用の仮想ページに書き込まれる。このデコードされたデータは、データ暗号鍵 Kd を用いて暗号化された後、ＲＡＭ１７内の共有物理ページに記録される。ＲＡＭ１７の共有物理ページに記録されたデータはデータ復号鍵 Kd を用いて復

号されて、ＤＲＭプロセス用の仮想ページに書き込まれる。プレーヤ（再生アプリケーション）プロセスはそのデータを読み出して再生する。

以下、このような保護データの共有を設定する手順について詳しく説明する。

まず、保護データを共有するために、共有されることになる保護データ領域を生成した生成元モジュールとその保護データ領域を生成元モジュールと共有する共有先モジュールを想定する。

生成元モジュールと共有先モジュールとがパッケージとして異なっている（すなわち Kc が異なる）にもかかわらず、共有先モジュールが生成元モジュールの保護データ領域を読み出すことができるようにするためには、図３２に示すようにして、生成元モジュールは共有先モジュールを認証しなければならない。その認証が正常に完了できた上で、ＧＴ１０内部において、生成元モジュールの保護データ領域の暗号鍵 Kd を共有先モジュールが利用できるようにＴＬＢおよびＴＲＢが設定されなければならない。

以下、図３２に沿って、モジュールの認証、並びに、保護データ復号鍵共有手順の設定手順について説明する。図３２において、括弧内の数字は、処理の順番を示し、以下の説明における手順の番号に対応する。また、図３２中の各記号の意味は次のとおりである。

RN：生成元モジュールが一時的に生成した乱数

KPacm：ソフトウェアモジュール向け認証局のルート公開鍵

Kacm：ソフトウェアモジュール向け認証局のルート秘密鍵

Kdp：共有先モジュールの秘密鍵

KPdp：共有先モジュールの公開鍵

C（KPdp, Kacm）：共有先パッケージの公開鍵証明書

Kc1:生成元モジュールのコード復号鍵

Kc2:共有先モジュールのコード復号鍵

（１）生成元モジュールが共有先モジュールに一時的に生成した乱数を渡す。

（２）共有先モジュールが、自身のプロセスが生成した仮想領域のＩＤとKc2 を KPgt で暗号化した値と乱数とを連結したデータに、そのデータのデジタル署名と署名に用いた秘密鍵に対応する公開鍵の証明書とを付加する。この証明書は PKI ライブラリ用認証局(CApki)、DRM 用認証局(CAdrm)あるいはアプリケーション向け認証局(CAap)などの信頼できる第三者が発行したものでなければならない。なお、この説明では、証明書には、復号鍵 Kc="YYYYYY"が KPgt で暗号化されているとする。

（３）共有先モジュールは、（２）で生成したデータを、生成元モジュールに一般のデータ共有／プロセス間通信などを利用して渡す。

（４）生成元モジュールは、署名をチェックし、その正当性を確認できた場合、受け付けた被許諾コード鍵 Authorized Code Key とアクセス権’PR’（耐攻撃モードで読み出し専用）と暗号・復号鍵 Kd を埋め込んだＧＴライセンスと指定仮想領域をパラメタとして、アクセス許諾命令(AUTHORIZE)を実行する。なお、この説明では、暗号・復号鍵 Kd="AAAAAA"がＧＴライセンスに埋め込まれているとする。

（５）アクセス許諾命令を実行した結果、共有先モジュールのＴＬＢにリンクしたＴＲＢ内の行に、ＧＴライセンスの内容が設定される。これにより、共有先モジュールが共有保護領域から読み出すデータは、鍵 Kd で正常に復号された状態でキャッシュされる。

この結果、図３２に示す例によれば、ＴＬＢの４行目には、物理ページ番号"002"へのアクセス権"PR"が設定され、このＴＬＢ内の行に対応するＴＲＢの４行目には、データ鍵 Kd="AAAAAA"が Key フィールドに設定される。これにより、生成元モジュールの保護データ領域（ＴＬＢの２行目に対応）を、共有先モジュールが、データ鍵 Kd="AAAAAA"を用いて復号して読み出す事が可能とな

る。

なお、異なるモジュール間で相互認証を行う場合には、手順（２）において共有先モジュールは、生成元モジュールの公開鍵または公開鍵で暗号化した対称鍵暗号法の鍵で、生成した送信メッセージを暗号化して、生成元モジュールに渡せばよい。これにより、このメッセージは生成元モジュール以外では復号できないこととなる。

なお、図１０に示すＴＬＢ内で、ＩＤが２である行とＩＤが７である行は、保護領域を共有した場合を例として示したものである。

図３３に、耐攻撃領域共有システムコールを呼び出した際の処理の手順を示す。この手順（Ｓ１１０からＳ１２３）は、システムコール時の入力パラメタが領域サイズの代わりに領域の先頭のアドレスであることをのぞいて、図２８に示す手順（Ｓ９０からＳ１０３）と同様であるため、説明を省略する。

なお、コンテンツ再生(Player)アプリケーションは、上述のローカルデータ共有方式を用いてデコーダ DRM４０からデータを得て、これを再生し、さらに編集することとしてもよい。編集し、その結果を記録する場合には、ＧＴライセンスにおいて指定されたアクセス条件に従わなければならない。また、再生アプリケーションはこれらの処理をＧＴ保護コードとして生成しなければならない。

再生時間や期限の制限を再生アプリケーションに強制するためにはセキュアタイマーが必要である。セキュアタイマーの構築は、ＧＴ１０の外に独立したハードウェア TRM として実現することしてもよい。または、ＧＴ１０が水晶発信機などを内蔵することが可能である場合、上述の定期割り込み間隔設定コマンドを用いて耐攻撃ソフトウェアとして構築することとしてもよい。いずれの場合も、タイマーがなんらかの理由で停止する場合を想定すると、外部の Trusted Secure Timer と同期をとる機能を持つ必要がある。

＜ＤＲＭソフトウェア構築モデル＞

以下、ＤＲＭソフトウェアモジュールの構築モデルに説明する。デコーダＤＲＭ３０、メディアＤＲＭ４０及びこれらのＤＲＭが用いるＰＫＩライブラリ２０内の暗号・復号ライブラリ、更には、ＴＲＭ化が必要なその他のアプリケーションは、ＧＴ１０上で保護されるＧＴ保護コードとして流通し、実行される。これらのモジュールは、ほぼ全文を暗文とすることによりＧＴ１０で保護される。

図３４に、ＤＲＭソフトウェアモジュールの構築モデルの一例を示す。このモデルでは、上記４つのモジュールが異なる開発メーカーで開発され、異なるコード暗号鍵 key (Kc)で暗号化されていることを想定している。図３４において、細い黒矢印は、暗号化されたライセンスキーのやり取りを示し、細い白矢印は、復号されたライセンスキーのやり取りを示し、太い黒矢印は、暗号化されたコンテンツのやり取りを示し、太い白矢印は、複合されたコンテンツのやり取りを示す。括弧内の数字は、手順の順番を示す。以下、図３４に沿って、コンテンツとそのライセンスキーのダウンロード、管理、再生手順について説明する。

（１）インターネット等のネットワークからダウンローダアプリケーションを介して、暗号化ライセンスキー及び暗号化コンテンツは記録媒体に記録される。

（２）ライセンスキーは暗号化された状態で、ダウンローダーを介してメディア DRM４０に転送される。

（３）メディア DRM４０内で復号されたライセンスキーは、後述の方法を用いて安全に管理され、暗号化された状態で記録媒体に記録される。ライセンスキーの再暗号化はＧＴ１０で保護された PKI 暗号ライブラリ２０を用いて実施される。

（４）ユーザの再生要求があった場合、メディア DRM４０は記録媒体からライセンスキーを取り出し、復号する。

（５）メディア DRM４０はデコーダ DRM３０を認証し、デコーダ DRM３０と共有するセッションキーを用いてライセンスキーを暗号化してデコーダ DRM３０に転送する。なお、この鍵の共有及び手順（７）の保護データの共有については＜保護データ領域の安全な共有＞において説明した。

（６）デコーダ DRM３０は記録媒体から取り出した暗号化コンテンツと、メディア DRM４０から得たライセンスキーとをパラメタとして PKI 暗号ライブラリ２０に復号処理を依頼する。

（７）復号されたコンテンツは共有保護データ領域を介して Player など再生アプリケーションに渡される。

（８）再生アプリケーションがコンテンツを再生する。

以下、図３５から図４０を用いて、上述の手順をより詳しく説明する。

まず、図３５及び図３６を用いて、メディアＤＲＭによって行われる処理について説明する。

まず、メディアＤＲＭ４０は、耐攻撃権限設定（SET_TR_RIGHTS）命令及びレジスタの封印（SEAL_REG）命令を実行する（Ｓ１３１及びＳ１３２）。続いて、メディアＤＲＭ４０は、自身に埋め込まれた秘密情報を取り出す（Ｓ１３３）。メディアＤＲＭ４０は、取り出された秘密情報に対応するＧＴライセンスが、ライセンスを記録するライセンスＤＢに格納されているか否か判定する（Ｓ１３４）。既にライセンスＤＢに格納されている場合（Ｓ１３４：あり）、Ｓ１３６に進み、まだライセンスＤＢに格納されていない場合（Ｓ１３４：なし）、メディアＤＲＭ４０は、そのＧＴライセンスをライセンスＤＢに設定した後（Ｓ１３５）、Ｓ１３６に進む。

Ｓ１３６において、メディアＤＲＭ４０は、ライセンス管理用の耐攻撃デー

タ領域を生成する。Ｓ１３６の処理について、詳しくは、図３６を用いて後述する。

ライセンス管理用の耐攻撃データ領域を生成できた場合（Ｓ１３７：NORMAL）、Ｓ１３８に進み、ライセンス管理用の耐攻撃データ領域を生成できなかった場合（Ｓ１３７：ERROR）、Ｓ１４２に進む。

Ｓ１３８において、メディアＤＲＭ４０は、ライセンス管理用の耐攻撃データ領域を初期化し（Ｓ１３８）、ライセンス受信要求、再生許諾要求又は終了要求を待つ。ライセンス受信要求を受けた場合（Ｓ１３９：あり）、メディアＤＲＭ４０はライセンスを受信して（Ｓ１４３）、Ｓ１３９にもどる。再生許諾要求を受けた場合、メディアＤＲＭ４０は再生許諾を行い（Ｓ１４４）、Ｓ１３９にもどる。終了要求を受けた場合（Ｓ１４１：あり）、レジスタ解放（RELEASE_REG）命令を実行し（Ｓ１４５）、処理を終了する。

ライセンス管理用の耐攻撃データ領域の生成に失敗した場合（Ｓ１３７：ERROR）、メディアＤＲＭ４０は、エラー表示を出力装置（不図示）に出力し（Ｓ１４２）、Ｓ１４５に進む。

次に、図３６を用いて、図３５のＳ１３６について説明する。ライセンス管理用耐攻撃データ領域を生成する際、まず、メディアＤＲＭ４０は、コード暗号鍵 Kd を用いてアクセス権 PRW のＧＴライセンスを生成する（Ｓ１５１）。続いて、メディアＤＲＭ４０は、耐攻撃領域取得システムコールを呼び出す（Ｓ１５２）。Ｓ１５２の処理については、上記のとおりである。

耐攻撃領域を取得する際にエラーが生じなかった場合（Ｓ１５３：なし）、途中に生成された署名を確認し（Ｓ１５４）、署名が正しければ（Ｓ１５５：一致）、正常にメインフローに戻る。耐攻撃領域を取得する際にエラーが生じた場合（Ｓ１５３：あり）、又は、署名が正しくない場合（Ｓ１５５：不一致）、エラーをメインフローに返す。

次に、図３７から図３９を用いて、デコーダＤＲＭによって行われる処理について説明する。まず、デコーダＤＲＭ３０は、耐攻撃権限設定（SET_TR_RIGHTS）命令及びレジスタの封印（SEAL_REG）命令を実行する（Ｓ１６１及びＳ１６２）。続いて、メディアＤＲＭ４０は、自身に埋め込まれた秘密情報を取り出す（Ｓ１６３）。

さらに、デコーダＤＲＭ３０は、復号されたコンテンツ用の共有保護データ領域を生成する（Ｓ１６４）。このＳ１６４について、詳しくは図３８を用いて後述する。

Ｓ１６４の処理が正常に行われた場合（Ｓ１６５：NORMAL）、デコーダＤＲＭ３０は、再生許諾要求、再生要求及び終了要求の何れかを受けるまで待つ。Ｓ１６４の処理が正常に行われなかった場合（Ｓ１６５：ERROR）、デコーダＤＲＭ３０は、出力装置（不図示）にエラーが生じた旨を表示し（Ｓ１６９）、Ｓ１７５に進む。

再生許諾要求を受けた場合（Ｓ１６６：あり）、デコーダＤＲＭ３０は、メディアＤＲＭ４０からコンテンツのライセンスキーを取得し（Ｓ１７０）、Ｓ１６６に戻る。

再生要求を受けた場合（Ｓ１６７：あり）、デコーダＤＲＭ３０は、コンテンツのライセンスキーを取得済みであるか否か判定する（Ｓ１７１）。コンテンツキーを取得済みである場合（Ｓ１７１：取得済）、デコーダＤＲＭ３０は暗号化ブロックを復号し、そのブロックを共有する処理を行う（Ｓ１７３）。Ｓ１７３についいて詳しくは図３９を用いて後述する。続いて、デコーダＤＲＭ３０は、再生アプリケーションに返却値を通知し（Ｓ１７４）、Ｓ１６６に戻る。Ｓ１７１の判定において、まだライセンスキーを取得していない場合（Ｓ１７１：なし）、出力装置（不図示）にエラー表示を行い（Ｓ１７２）、Ｓ１７５に進む。

終了要求を受けた場合（Ｓ１６８：あり）、Ｓ１７５に進み、デコーダＤＲＭ３０は、レジスタ解放（RELEASE_REG）命令を実行し、処理を終了する。

次に、図３８を用いて、図３７のＳ１６４について詳しく説明する。復号されたコンテンツ用の共有保護データ領域を生成するために、まず、デコーダＤＲＭ３０は、デコーダＤＲＭ３０によってデコードされたコンテンツを記録するための耐攻撃領域を取得するために耐攻撃領域取得システムコールを呼び出す（Ｓ１８１）。この処理については既に説明した。耐攻撃領域が生成できた場合（Ｓ１８２：NORMAL）、Ｓ１８４に進む。そうでない場合（Ｓ１８２：ERROR）、エラー表示を出力装置（不図示）に出力させ（Ｓ１８３）、メインフローに復帰する。

Ｓ１８４において、デコーダＤＲＭ３０は、ＧＴ１０に、暗号化されたコンテンツを復号するために用いるＰＫＩ暗号ライブラリを認証させる。なお、この認証手順は、＜保護データ領域の安全な共有＞において、図３２を用いて説明した認証手順と同様である。

Ｓ１８４の認証が正常に行われた場合（Ｓ１８５：NORMAL）、Ｓ１８７に進む。Ｓ１８４の認証が正常に行われなかった場合（Ｓ１８５：ERROR）、エラー表示を出力装置（不図示）に出力させ（Ｓ１８６）、メインフローに復帰する。

Ｓ１８７において、デコーダＤＲＭ３０は、耐攻撃領域共有システムコールを呼び出す。Ｓ１８４の処理が正常に行われた場合（Ｓ１８８：NORMAL）、メインフローに復帰する。これにより、デコーダＤＲＭ３０とＰＫＩライブラリが、耐攻撃領域に書きこまれた保護データを共有することができるようになる。Ｓ１８４の処理が正常に行われなかった場合（Ｓ１８８：ERROR）、エラー表示を出力装置（不図示）に出力させ、メインフローに復帰する。

次に、図３９を用いて、図３７のＳ１７３についてより詳しく説明する。ま

ず、デコーダＤＲＭ３０は、再生アプリケーションは既に認証されているか否か判定する（Ｓ１９１）。既に認証されている場合（Ｓ１９１：認証済）、Ｓ１９６に進む。まだ認証されていない場合（Ｓ１９１：未認証）、Ｓ１９２からＳ１９５が行われる。この処理は、図３８のＳ１８４からＳ１８８と同様であるため、説明を省略する。Ｓ１９２からＳ１９５の処理においてエラーが発生した場合（Ｓ１９３及びＳ１９５で ERROR）、エラー表示を出力装置（不図示）に出力させ（Ｓ１９８）、メインフローに復帰する。Ｓ１９２からＳ１９５が正常に行われた場合、再生アプリケーションとデコーダＤＲＭ３０とは耐攻撃領域を共有し、再生アプリケーションは、デコーダＤＲＭ３０によって耐攻撃領域に書きこまれたコンテンツを読み出すことができるようになる。

Ｓ１９６において、デコーダＤＲＭ３０は、暗号化されたコンテンツをＰＫＩ暗号ライブラリ２０を用いて復号する。復号されたコンテンツは、共有化された耐攻撃領域に書き込まれる。復号が正常に行われた場合（Ｓ１９７：NORMAL）、メインフローに復帰する。復号が正常に行われなかった場合（Ｓ１９７：ERROR）、Ｓ１９８に進む。

次に、図４０を用いて、再生アプリケーションによって行われる処理について説明する。

まず、再生アプリケーションは、耐攻撃権限設定（SET_TR_RIGHTS）命令及びレジスタの封印（SEAL_REG）命令を実行する（Ｓ２０１及びＳ２０２）。続いて、再生アプリケーションは、自身に埋め込まれた秘密情報を取り出す（Ｓ２０３）。続いて、再生アプリケーションは、デコーダＤＲＭ３０に認証を要求する（Ｓ２０４）。この認証要求に基づいて、上記Ｓ１９２が行われる。

認証が正常に行われた場合（Ｓ２０５：NORMAL）、再生アプリケーションは、再生要求、その他の要求又は終了要求を受けるのを待つ。認証が正常に行われなかった場合（Ｓ２０５：ERROR）、再生アプリケーションは、エラー表示を

出力装置（不図示）に出力させ（Ｓ２１４）、Ｓ２１６に進む。

再生要求を受けた場合（Ｓ２０６：あり）、再生アプリケーションは、デコーダＤＲＭ３０に暗号ブロックを復号するよう要求する（Ｓ２１０）。デコーダＤＲＭ３０からの返却値が正常な場合（Ｓ２１１：NORMAL）、再生アプリケーションは、そのブロックを再生する（Ｓ２１２）。コンテンツの再生が終了するまで（Ｓ２１４：Ｎｏ）、Ｓ２１０からＳ２１２は繰り返される。コンテンツの再生が終了すると（Ｓ２１４：Ｙｅｓ）、Ｓ２０６にもどる。

再生要求以外の要求を受けると（Ｓ２０７：あり）、再生アプリケーションはその要求を実行し（Ｓ２０８）、Ｓ２０６にもどる。終了要求を受けると（Ｓ２０９：あり）、再生アプリケーションはＧＴ１０にレジスタ解放（RELEASE_REG）命令を実行させ（Ｓ２１６）、処理を終了する。

＜秘密情報の管理＞

証明書を発行された公開鍵暗号法の公開鍵に対応する秘密鍵 Kdrm など、DRM コードパッケージ開発者が生成する秘密情報を維持・管理する方式を図４１に示す。図４１において、細い矢印はデータの流れを示し、太い白矢印は、鍵等の埋め込みを示す。また、括弧内の数字は処理の順番を示し、以下の説明における手順の番号に対応する。

特に公開鍵証明書に対応する秘密鍵は、CRL の指定時などに、特定の GT と特定の DRM のセットのみを停止する必要性から、本方式を用いる必要がある。本方式の手順はおおよそつぎのようになる。

（１）開発者が生成した秘密鍵（Kdrm など）は、クラス・個別いずれの場合も開発者が生成した対称鍵暗号法の鍵 Kprd で暗号化され、パッケージ内に埋め込まれる。

（２）ＧＴライセンス化された、Kprd とアクセス条件’PR’とは、耐攻撃モードでのみ読み出し可能な形態で保護コード内に埋め込まれる。

（３）コード全体は、Kc で暗号化され DRM として保護パッケージ化される。

（４）Kc は KPgt で暗号化され、DRM ライセンスとして DRM パッケージに入れられる。

（５）DRM 実行時に VHTRM からこの GT ライセンスはＧＴ１０に投入され、ＧＴ１０内部で Kprd が読み出される。

（６）Kprd で Kdrm は復号され、利用される。

これにより、開発者が指定したＧＴ上の指定プロセス以外のプロセスは、秘密鍵 Kdrm を読むことができなくなる。特定のＧＴが破られた場合には、破られたＧＴの公開鍵暗号法の公開鍵証明書のみを失効すればよい。従って、同じ DRM 向けのものでも、他の GT 向けの証明書（Cdrm2）は正当に利用することができる。

本方式は DRM の秘密鍵のみでなく、他の保護パッケージの秘密鍵を管理する方法としても利用できる。

＜ＵＤＡＣライセンス管理＞

保護コンテンツと UDAC ライセンスに関する一般情報の管理方式は、UDAC-MB および UDAC-LA の方式を引き継ぐこととしてもよい。図４２にライセンス管理のためのメモリアクセスの方式を示す。

図４２に示すように、コンテンツ復号鍵などのライセンスの秘密情報は、ＲＡＭ１７内の保護データ領域に記録されて管理される。その保護データ領域内の情報は、ファイルとしてストレージやフラッシュメモリなどに記憶されることとしてもよい。秘密情報を記録する際にデータ暗号鍵 Kd で暗号化することにより、ＣＰＵ（ＧＴ１０）の再起動時にもライセンスの秘密情報が安全に保護された状態で利用できる。このデータ暗号鍵 Kd はＴＲＢ暗号鍵 Ktrb で暗号化された状態で、暗号化キーテーブルの行として一般の外部記憶装置に保持されることしてもよい。

なお、Ktrb と Ktlb の耐攻撃能力を高めるためには、一定の期間を置いて Ktrb と Ktlb の内容をリフレッシュする必要がある。リフレッシュの前にはライセンス情報をすべて、耐攻撃プロセスが生成した一時キーで暗号化してバックアップしなければならない。リフレッシュによる Ktrb 及び Ktlb 変更のあとに、ページテーブルと暗号化キーテーブルを再構築する。全ライセンス情報を復号し、再構築した耐攻撃領域にリストアすればよい。

上記の秘密鍵管理が前提としてＣＲＬの管理を行う事も可能である。また、ライセンスごとの CRL 制御機能は上記のライセンス管理機能内に持つこととしてもよい。１つの DRM パッケージの証明書は、ＧＴ１０が持っている証明書の数だけ発行されることを原則とする。DRM パッケージ自体を失効する場合にはそれらの証明書のすべてが失効される。また特定の GT を失効する場合には、その GT 用に発行した DRM パッケージ証明書のみが失効される。

プログラムコンテンツを超流通し、UDAC ライセンスを扱う場合、DRM ソフトウェアを利用せず、簡易版超流通機能として、GT ライセンスをそのまま利用してもよい。また、GT 上に構築した DRM ソフトウェアを用いて、UDAC-MB や PI ライセンスとして処理してもよい。 DRM ソフトウェアを用いる場合、UDAC-MB/PI ライセンスのうち、基本アクセス権(PR, X, PRW, PWX)のみは DRM 内で GT ライセンスに変換し、CPU のアクセス制御命令(AUTHORIZE)を用いて強制する。その他のアクセス条件については、GT 保護 DRM 内で強制処理を行う。

当然ながら、UDAC のみでなく、今日のソフトウェア TRM を用いた各社の DRM も GT 保護化することで、ハードウェア TRM レベルの耐攻撃性を持つことができる。

＜変形例＞

上記において、ＧＴライセンスは移動しない事として説明したが、DRM 間のライセンス移動機能の実現する事も可能である。同じＧＴ上で実行される２つ

のＤＲＭプロセス間でのライセンス移動・許諾は、＜保護データの安全な共有＞で認証された共有耐攻撃領域を介して実現することとしてもよいし、UDAC-MB や UDAC-PI などのプロトコルを利用して実現する事としても良い。但し、ライセンスの移動機能を実装する場合、再送攻撃による不正コピーを防止するためには、ＴＲＭ内に、以下の UPL (Unreplicatable Private Locker：複製不能プライベートロッカー)にライセンス管理用秘密情報を格納しなければならない。この UPL を GT のみで実現したい場合には、TRB.uo（ＴＲＢ内の uo フィールドの値）および TRB.cl（ＴＲＢ内の cl フィールドの値）を両方とも on にした耐攻撃領域にライセンス管理用秘密情報を格納することでこれが可能になる。

ただし、GT の電源を切断後も UPL を用いて管理しているライセンスを維持する必要がある場合には、ライセンス管理用秘密情報を格納する UPL の領域だけでも不揮発性記憶素子にし、 Permanent UPL（半永久 UPL）にする必要がある。耐攻撃空間の一部を Permanent UPL に割り当てる方法については、ここでは特定しない。

GT の外部に TRM 化した UPL を実現するためには、UPL は UDAC-MB などの TRM 認証プロトコルを UPL 内に実装する必要がある。外部に実現する Permanent UPL としては UDAC を実装した Secure MMC などを利用することができる。

次に、第２実施形態について説明する。第１実施形態において、コードブロック及びデータブロックは、平文又は暗文であった（ebim＝０又は１）。第２実施形態によれば、ブロックは、暗文及び平文更には他の情報の組合せであってもよい。

以下、図４３を用いて第２実施形態に係わるＧＴを実現するＣＰＵの構成について説明する。

なお、図４３において、図３に示す第１実施形態と同様の部分は省略されている。図４３に示すように、第２実施形態に係わるＧＴ１００は、第１実施形態

にかかわる構成に加えて、さらに、保護ブロックセレクタ１０１及び１０４、ハッシュチェッカ１０２、及びハッシュエンジン１０３を備える。

保護ブロックセレクタ１０１は、キャッシュ１１又は１２から出力されるコードブロック又はデータブロックが、保護されるべきブロックであるか否かをebim の値に基づいて識別する。出力されたブロックが保護されるべきブロックである場合、保護ブロックセレクタ１０１は、そのブロックをハッシュエンジン１０３に出力する。ハッシュエンジン１０３はそのブロックのハッシュを生成し、ハッシュが生成された後に、暗号ブロック１６はそのブロックを暗号化し、ＲＡＭ１７に出力する。一方、キャッシュ１１又は１２から出力されたブロックが保護されるべきブロックではない場合、保護ブロックセレクタ１０１は、そのブロックをＲＡＭ１７に出力する。

また、保護ブロックセレクタ１０４は、ＲＡＭ１７から出力されるコードブロック又はデータブロックが、暗号化されているブロックであるのか平文のブロックであるのか識別する。出力されたブロックが暗号化されているブロック、つまり保護ブロックである場合、保護ブロックセレクタ１０４は、その保護ブロックを暗号エンジン１６に出力する。暗号エンジン１６は、その保護ブロックを復号し、復号されたブロックをハッシュエンジン１０３に出力する。ハッシュエンジンは、そのブロックのハッシュを生成し、ハッシュチェッカ１０２に出力する。ハッシュチェッカ１０２は、生成されたハッシュを確認し、ハッシュが正しい事を確認できた場合、そのブロックをキャッシュ１２又は１３に出力する。一方、ＲＡＭ１７から出力されたブロックが保護ブロックではない場合、保護ブロックセレクタ１０４は、そのブロックをキャッシュ１２又は１３に出力する。

なお、図４３において、保護ブロックセレクタ１０１及び１０４を備えることとしたが、保護ブロックセレクタ１０１及び１０４を１つの保護ブロックセ

レクタとして構成することとしてもよい。これにより、回路を小型化することが可能となる。

次に、第２実施形態に係わるＴＬＢ内のフィールドについて説明する。図５に示すＴＬＢには、ebim フィールドが含まれる。第１実施形態によれば、この ebim の値は０又は１であった。第２実施形態によれば、この ebim の取りうる値として、更に２から７が追加される。以下、ebim の値に応じるブロックの構造について説明する。

ａ）ebim＝０の場合、平文のみ（ダミーライセンス、第１実施形態と同様）

ｂ）ebim＝１の場合、暗文のみ（第１実施形態と同様）

ｃ）ebim＝２の場合、保護ブロック開始識別コード使用

ｄ）ebim＝３の場合、同上かつ暗号化ブロックのみ

ｅ）ebim＝４の場合、署名付平文ブロックのみ。
キャッシュロード時署名確認必須

ｆ）ebim＝５の場合、ハッシュ付暗号化保護ブロックのみ。
復号後ハッシュ確認必須

ｇ）ebim＝６の場合、保護ブロック開始識別コード使用。
署名／ハッシュ確認必須

ｈ）ebim＝７の場合、同上かつ暗号化ブロックのみ。ハッシュ確認必須

すなわち、本フィールドの各ビットは、次のような意味を持つ。

ビット０：暗号化フラグ。このフラグがオンである場合、ブロックが暗号化されている事を示す。

ビット１：保護ブロック開始識別コードフラグ。このフラグがオンである場合、ブロックに保護ブロック開始識別コードが付加されていることを示す。個フラグがオフである場合、ＧＴ１０は、保護ブロック開始識別コードを認識する必要はない。

ビット２：ハッシュ付加・確認フラグ。このフラグがオンである場合、データをＧＴ１０の外に掃出する際に、ＧＴ１０は、そのデータブロックにハッシュを付加する。また、コードやデータをＧＴ１０内に取り込む際には、ブロックに付加されたハッシュ値を確認する。このフラグがオフである場合、ブロックにハッシュを付加したり確認したりする必要はない。

これらの ebim の値は、第１実施形態と同様に、ＧＴライセンスに含まれる被許諾コード鍵 ACgt 内の ebim フィールドの値に基づいて設定される。

次に、図４４を用いて第２実施形態におけるブロックの構造について説明する。ＧＴ１０によれば、保護コードブロックと保護データブロックの構造を同じ形式にしても良い。これにより、ＣＰＵ内の回路リソースの利用率を向上させることが可能となる。図４４に示すように、開発者や創作者によって生成されたブロックは、乱数、一般命令やデータを含み、必要な場合さらにハッシュを含む。ebim が１又は５である場合、このブロックが暗号化されることにより、保護ブロックが生成される。ebim が２、３、６又は７である場合、このブロックが暗号化された後に、保護ブロックヘッド識別コードが平文で先頭に付加されることにより。保護ブロックが生成される。保護ブロックヘッド識別コードは、ブロックが保護ブロックであるか否かを示す暗号化フラグ及び、ブロックの最後にハッシュが付加されているか否かを示すハッシュフラグを含む。保護ブロックヘッド識別コード及び暗号化された乱数部分は、保護ブロック開始命令を構成する。

ＲＡＭ１７上の保護ブロックは、ＧＴ１０内で復号されることにより、平文の一般命令やデータが取得される。ebim が１又は５である場合、乱数部分やハッシュ部分は、コードやデータのアドレスがかわらないように、ＮＯＰ（No OPeration）命令に変換されてから、命令キャッシュ１２又はデータキャッシュ１３にロードされる。なお、ebim が１である場合は、第１実施形態と同じで

ある。ebim が２、３、６又は７である場合、乱数部分、ハッシュ部分に加えて保護ブロックヘッド識別コードも、ＮＯＰ（No OPeration）に変換されてから、命令キャッシュ１２又はデータキャッシュ１３にロードされる。

なお、プロセッサコア１１上の作業データ等を暗号化した場合の保護ブロックの構造も、開発者又は創作者によって生成されたブロックを暗号化した場合の保護ブロックと同様の構造である。

図４５に、ebim が４である場合のブロックの構造を示す。図４５に示すように、ebim が４である場合、ＲＡＭ上のブロックは、平文のコード又は平文のデータに署名が付加された構造となっている。命令キャッシュ１２又はデータキャッシュ１３上にブロックをロードする際には、署名はＮＯＰ命令に変換される。なお、署名は、コード又はデータのハッシュ（ＳＨＡ－１など）を暗号鍵 Kc 又は Kd で暗号化した値とすることとしてもよい。なお、Kc 及び Kd は、ＧＴライセンスで指定され、ＴＲＢの暗号鍵フィールド（TRB.key）に設定されている。

次に、第２実施形態に係わるＧＴによって行われる処理について説明する。基本的な動作は、第１実施形態と第２実施形態とは同様であるため、以下の説明において、第２実施形態において更に付加された構成によって行われる処理に重点をおいて説明する。

まず、図４６を用いて、保護ブロックセレクタ１０４によって行われる処理について説明する。

まず、保護ブロックセレクタ１０４は、ＲＡＭ１７（外部メモリ）からブロックのロード要求が出されるのを待つ（Ｓ２２１　）。ＲＡＭ１７からブロックのロード要求が出されると（Ｓ２２１：要求）、保護ブロックセレクタ１０４は、予測されるブロックのアドレスと対応する行の ebim フィールドの値をＴＬＢから読み込む（Ｓ２２２）。

保護ブロッセレクタ１０４は、ebim の第１ビットがオンであるか否か判定する（Ｓ２２３）。ebim の第１ビットは、ブロックに保護ブロック開始命令が付加されているか否かをしめす。ebim の第１ビットがオンである場合（Ｓ２２３：ｏｎ）、そのブロックには保護ブロック開始命令が付加されている。保護ブロックセレクタ１０４は、そのブロックの先頭に付加されている保護ブロック開始命令を読み出し（Ｓ２２４）、その保護ブロック開始命令において暗号化フラグがオンであるか否か判定する（Ｓ２２５）。暗号化フラグは、暗号化フラグがオンである場合（Ｓ２０５：ｏｎ）、Ｓ２２９に進む。Ｓ２２９において、保護ブロックセレクタ１０４は、そのブロックを暗号エンジン１６に出力し、Ｓ２０１に戻る。この場合、そのブロックは、復号された後にキャッシュ１２又は１３に転送される。

暗号化フラグがオフである場合（Ｓ２２５：ｏｆｆ）、保護ブロックセレクタ１０４は、さらに、ebim の第２ビットがオンであるか否か判定する（Ｓ２２６）。ebim の第２ビットは、そのブロックを転送する際にハッシュの確認が必要であるか否かを示す。ebim の第２ビットがオンである場合（Ｓ２２６：ｏｎ）、処理はＳ２０９に進む。ebim の第２ビットがオフである場合（Ｓ２２６：ｏｆｆ）、保護ブロックセレクタ１０４はそのブロックをキャッシュ１２又は１３に転送する（Ｓ２２７）。

Ｓ２２３の判定において、ebim の第１ビットがオフである場合（Ｓ２２３：ｏｆｆ）、保護ブロッセレクタ１０４は、更に ebim の第０ビットがオンであるか否か判定する（Ｓ２２８）。ebim の第０ビットは、そのブロックを暗号化することが必要であるか否かを示す。ebim の第０ビットがオンである場合（Ｓ２２８：ｏｎ）、処理はＳ２２９に進む。ebim の第０ビットがオフである場合（Ｓ２２８：ｏｆｆ）、保護ブロックセレクタ１０４は、さらに ebim の第２ビットがオンであるか否か判定する（Ｓ２３０）。ebim の第２ビットがオンで

ある場合（Ｓ２３０：ｏｎ）、処理はＳ２２９に進む。ebim の第２ビットがオフである場合（Ｓ２３０：ｏｆｆ）、保護ブロックセレクタ１０４は、そのブロックをキャッシュ１２又は１３に転送し（Ｓ２３１）、Ｓ２２１に戻る。

続いて、図４７を用いて、ハッシュエンジン１０３によって行われる処理について説明する。

まず、ハッシュエンジン１０３は、ハッシュ要求のイベントが発生することを待つ（Ｓ２４１）。イベントが発生すると、ハッシュエンジン１０３は、ハッシュの要求が行われたブロックを読み込む（Ｓ２４２）。さらに、ハッシュエンジン１０３は、そのブロックのアドレスと対応する ebim を読み込み、その ebim の第２ビットがオンであるか否か判定する（Ｓ２４３）。ebim の第２ビットがオンである場合（Ｓ２４３：ｏｎ）、ハッシュエンジン１０３は、そのブロックのハッシュを生成し（Ｓ２４４）、そのブロックと生成したハッシュの内容を次の回路ブロックに転送し（Ｓ２４５）、Ｓ２４１に戻る。一方、ebim の第２ビットがオフである場合（Ｓ２４３：ｏｆｆ）、ハッシュエンジン１０３は、ハッシュを生成しないでそのブロックを次の回路ブロックに転送し（Ｓ２４６）、処理はＳ２４１に戻る。

続いて、図４８を用いて、ハッシュチェッカ１０２によって行われる処理について説明する。

まず、ハッシュチェッカ１０２は、ハッシュ要求のイベントが発生する事を待つ（Ｓ２５１）。イベントが発生すると、ハッシュチェッカ１０２は、要求が行われたブロックを読み込む（Ｓ２５２）。続いて、ハッシュチェッカ１０２は、そのブロックのアドレスと対応する ebim を読み込み、その ebim の第２ビットがオンであるか否か判定する（Ｓ２５３）。ebim の第２ビットがオンである場合（Ｓ２５３：ｏｎ）、ハッシュチェッカ１０２は、ハッシュエンジン１０３によって生成されたハッシュと、そのブロックに付加されているハッシ

ュとを比較する（Ｓ２５４）。比較の結果、２つのハッシュが一致しない場合（Ｓ２５４：不一致）、プロセッサコア１１に、ハッシュ不一致例外が発生したことを通知し（Ｓ２５５）、Ｓ２５１に戻る。２つのハッシュが一致した場合（Ｓ２５４：一致）、ハッシュチェッカは、そのブロックを次の回路ブロックに転送し（Ｓ２５６）、Ｓ２３１に戻る。

次に、第２実施形態の変形例について説明する。第２実施形態では、ebim に基づいて、保護ブロックセレクタは、ブロックを選択するとして説明した。この方式以外の方式を以下に例示する。

１．実行ファイルのヘッダに暗号化ブロックビットマップを埋め込む方法。保護ブロックセレクタは、このビットマップを先に読み込んで、ブロックを一般ラインに出力するか、復号ラインに出力するかを判断する。

２．コードの先頭を平文にし、この先頭のコードにおいて、何ブロックの平文のあと、何ブロックの保護コードがきてこれを繰り返すかを指定し、ＧＴ１００は、最初にこのコード（インストラクション）を実行する。このコードは、途中で変更することも可能である。この方式によれば保護コードブロックセレクタの機能を縮小できる。アドレスの下位ビットだけで、保護コードをセレクトすることができれば、さらにセレクタの機能を縮小させることが可能になる。

次に、第３実施形態について説明する。第１及び第２実施形態によれば、保護コードや保護データがキャッシュ内に入る時には、乱数を変換した結果として得られたＮＯＰがキャッシュライン長ごとに存在することになる。これにより、キャッシュ内のリソースを無駄に使用するという問題が生じる。第３実施形態は、その問題を解決する技術にかかわる。第３実施形態によれば、以下の方法１から３によってキャッシュのリソースを有効に利用することを可能となる。

方法１）乱数長とブロック数の積に、仮想領域長を加算した長さを、物理領

域長とする。（仮想領域長＋乱数長×ブロック数＝物理領域長）。ＲＡＭ１７上に仮想ページの大きさにはみ出す部分を持つことにより、はみ出した部分、つまり乱数を変換したＮＯＰが、キャッシュ内に記録されないようにする。

方法２）乱数長とブロック数の積に等しい長さを持つ領域を、ＮＯＰ専用の物理領域として設定する。

方法３）キャッシュ内にＮＯＰを記録しない。図４９に、本方法によるＮＯＰの処理を示す。図４９に示すように、ＲＡＭ１７における、保護コードブロック又は保護データブロックは、暗号化されている。なお、場合によっては、保護コードブロック又は保護データブロックは、保護ブロックヘッド識別コードを含む。この保護コードブロック又は保護データブロックがＧＴ１０内で復号されると、一般命令（又は一般データ）、乱数及びハッシュを含むブロックが得られるが、キャッシュ内には、このうちの一般命令又は一般データが記録され、ＮＯＰが記録されるべき仮想アドレスのデータは記録されない。コードがＮＯＰの仮想アドレスにアクセスした場合、ＮＯＰをコードに返す。なお、仮想メモリに記憶されるＮＯＰをＯＳ等から読むことができるようにしても良いし、読むことができないようにしても良い。

次に、第４実施形態について説明する。第４実施形態は、上記において説明したＧＴ１０を２以上備えるＣＰＵ、つまりマルチＣＰＵにおいて保護プロセスを動作させる場合に係わる。

マルチＣＰＵで保護プロセスを動作させる場合、以下のような場合に備える必要がある。

１．コード復号鍵 Kc を１つだけ持つ保護プロセスが、複数のスレッドを複数のＣＰＵ上で平行に実行する場合

２．スヌープ機能による保護コードや保護データなど、キャッシュ内容の自動同期に対応する。

図５０に、ＧＴ１０Ａ及びＧＴ１０Ｂを備えるマルチＣＰＵの構成をしめす。図５０に示すように、ＧＴ１０Ａ、ＧＴ１０Ｂ及びＲＡＭ１７が、バスを介して接続されている。ＧＴ１０Ａ及びＧＴ１０Ｂは、それぞれキャッシュを備える。複数のスレッドをＧＴ１０Ａ及びＧＴ１０Ｂによって並行に実行する場合、実行に先立って、ＧＴ１０Ａ及びＧＴ１０Ｂは、ＤＣＴＰ（Digital Transmission Content Protection）の完全認証（Full Authentication）等の相互認証を行う事によってセッション鍵 Ksnp を得る。ＧＴ１０Ａ及びＧＴ１０Ｂは、秘密情報、保護コード及び保護データを交換する場合、セッション鍵 Ksnp を用いる。これにより、ＧＴ１０Ａ及びＧＴ１０Ｂは Ktrb や Kc、Kd 等を安全に交換することが可能となる。ＧＴ１０Ａ及びＧＴ１０Ｂは、セッション鍵 Ksnp を用いてキャッシュ内のデータを暗号化し、互いに同期転送することにより、キャッシュ内容の同期を実現する。

次に、第４実施形態について説明する。従来のコンパイラやアッセンブラを用いて作成したプログラムコードオブジェクトは、ＧＴ保護関連の情報を持たない。これを図５１に示すような方式でＧＴでの保護コード実行形式、さらに SCDF（超流通コンテンツ形式：Super Content Distribution Format）に変換することができる。

図５１に、コードオブジェクトから保護コード実行形式を出力するツール群の例を示す。図５１では、保護コード実行形式とライセンスを生成し、さらに実行形式を超流通実施のために SCDF 生成ツールを通して SCDF データを生成する例を提案している。

図５１に示すように、ＳＣＤＦ生成ツールは、リンカ前処理部、リンカ、保護コード実行形式生成部、及びＳＣＤＦ作成部を備える。まず、コードオブジェクトは、リンカ前処理部によって、複数のブロックに分割され、それぞれのブロックにＮＯＰ命令が追加される。続いて、リンカは、アドレス解決を行う。

さらに、リンカの後、保護コード実行形式生成部は、コード暗号鍵を用いて各ブロックを暗号化することにより保護コード実行形式を生成する。一方、ＧＴライセンス生成部は、前記コード暗号鍵を含み、前記秘密鍵と対になる公開鍵によって暗号化されたライセンスを生成する。

次に、第５実施形態について説明する。ＧＴ１０による保護を実現するためには、ＧＴ１０と、ＧＴ１０で保護されるＤＲＭソフトウェアモジュールとが必要である。しかし当初はＧＴも普及しておらず、ＯＳもＧＴ１０の機能をサポートしていないため、次のようなシナリオで普及を推進する必要がある。

（１）当初はソフトウェア TRM により、CPU 拡張部分のハードウェア命令エミュレータを開発し、従来のＣＰＵ向けのＤＲＭソフトウェアをエミュレートする。これにより、従来のＣＰＵ向けのＤＲＭソフトウェアも、ＧＴ１０上で保護ソフトウェアとして稼動することが可能となる。

（２）DRM 部分は既存 DRM モジュールを流用する。

（３）ＯＳがＧＴ１０の機能をサポートするまでは耐攻撃空間管理、保護コンテキスト切り替え管理やDRM などをＯＳパッケージの外に持つ必要がある。

また、当初はアプリケーションとデコーダ DRM は一つのパッケージにし、Kc 及び Kd は同じとすることとしてもよい。

以下、上記各実施形態において説明したＧＴの応用例について、第６実施形態から第１２実施形態として説明する。

ＧＴの応用例について様々な例が挙げられるが、ここでは、例として、パーソナルコンピュータ、ワークステーション、ＰＤＡ（Personal Digital Assistance）、携帯電話（簡易型携帯電話を含む）、スマートカード等のＩＣカード、ＲＦＩＤメディア、ＡＶ（Audio Visual）機器、ＧＲＩＤ計算、ロボット等を例として挙げ、これらについて説明する。

まず、第６実施形態として、パーソナルコンピュータ及びワークステーショ

ンへのＧＴの応用について説明する。図５２に、ＧＴ１０又は１００を、パーソナルコンピュータ又はワークステーションに実装した場合のシステム構成例を示す。図５２に示すように、マザーボードにＧＴ１０又は１００が搭載されている。システムコントローラには、ＵＳＢ（Universal Serial Bus）コントローラ、ＩＤＥ（Integrated Drive Electronics）コントローラ、ＩＥＥＥ１３９４コントローラ、ビデオコントローラ及びオーディオコントローラ等が内蔵されている。

図５２において、デジタルコンテンツを記録するメモリカード、ＩＣカード、磁気ディスク、光磁気ディスク、デジタル多用途ディスク等の記録媒体には、メディアＤＲＭチップが埋め込まれている。このメディアＤＲＭチップは、ＧＴ１０又は１００上で保護されるモジュールである。このように構成することにより、特別に暗号・復号専用チップを組み込むことなく、ＧＴをコンピュータに採用することにより、ハードウェアＴＲＭと同じくらい強力に、デジタルコンテンツを保護することが可能となる。なお、図５２において、システムコントローラとして North Bridge が記載され、インタフェースとしてＵＳＢやＩＤＥが記載され、シリアルバスとしてＩＥＥＥ１３９４が記載されているが、システム構成を限定する趣旨ではない。

さらに、ソフトウェア開発・追加のみで、個人認証、TCPA、電子財布、個人情報保護、Trusted GRID Computing などをハードウェア TRM と同程度に強力なセキュリティレベルで実現する。

また、ＧＴ保護ソフトウェアとして作成された電子投票ソフトウェアをＰＣ等にロードする事により、ＰＣから電子投票を行うことが可能となる。また、ＧＴ保護ソフトウェアとして作成されたファイル管理ソフトウェアをＰＣ等にロードする事により、セキュアファイルシステムを構成することが可能となる。

次に、第７実施形態として、ＰＤＡや携帯電話等のモバイル機器への応用に

ついて説明する。

PC の TCPA (Trusted Computing Platform Alliance)機能についは、従来方式では別途特別なハードウェア装置を PC に接続する必要があったが、GT をＰＣに搭載すれば、その PC 上のソフトウェアで開発することのみで、この機能を実現できる。

携帯電話の端末認証機能も、同様に従来方式より強力なセキュリティ強度で実現できる。例えば、携帯電話のＳＩＭ (Subscriber Identity Module) カードを交換する機能は、ＧＴを実装する携帯電話間で保護データを交換するソフトウェアを用いる事により、より安全に実現される。

携帯電話やＰＤＡ (Personal Digital Assistance) などのモバイル製品にＧＴを適用する場合にも、特別に暗号・復号専用チップを組み込むことなく、ハードウェア TRM レベルの強力なコンテンツ保護を実現することができる。さらに、

もちろん、ソフトウェアを追加すれば、携帯電話やＰＤＡに、個人認証機能、クレジットカード機能、プリペイドカード機能、個人情報保護、機能などを与える事ができ、これらの機能はハードウェア TRM と同等の強力なセキュリティレベルで実現される。

次に、第８実施形態としてＩＣカードや RFID モジュールなどのセキュリティカードやモジュールへの応用について説明する。

ＩＣカードについては従来の方式ではＩＣカード内のセキュリティ機能をカスタマイズするごとに TRM 化を施した個別のチップ生産を行う必要があった。しかも、個別のチップごとにハードウェア TRM の評価基準について審査を行う必要があった。

しかし、本実施形態によれば、ＧＴに保護すべきセキュリティ機能を後からファームウェアとして追加するだけで、ハードウェア TRM と同等のレベルの追

加セキュリティ強度を持つＩＣカードを作成することができる。ＩＣカードのセキュリティ評価についても、ＧＴに追加したファームウェアについてのみ実施すればよい。

ＩＣカード、RFID モジュールなどのセキュリティカードやモジュールにＧＴ１０又は１００を実装すれば、カスタマイズしたチップごとに TRM 化を施す必要はなく、CPU 部分のみ TRM 化を施しておけばよい。これで特別に暗号・復号専用チップなどを組み込むことなく、ハードウェア TRM レベルの強力なメディア DRM を実現できる。なお、ソフトウェア追加のみで個人認証機能、クレジットカード機能、プリペイドカード機能などをハードウェア TRM レベルの強力なセキュリティで実現することも可能である。

さらに、CPU と比較して GT の寿命が短い場合に、GT のコア制御部分のみを交換できることとしてもよい。なお、ＧＴコア制御とは、ＴＲＭ、ＴＬＢ等にかかわる部分をいう。但し、この場合、CPU とＩＣカードを超高速バスで接続する必要がある。

次に、第９実施形態としてＡＶ機器への応用について説明する。

ＧＴをＡＶ機器に搭載する場合、カスタマイズしたチップごとに TRM 化を施す必要はなく、ＣＰＵ部分のみ TRM 化を施しておけばよい。これで特別に暗号・復号専用チップを組み込むことなく、ハードウェア TRM と同程度の強度を持つ強力な DRM を実現することができる。また、さらに、ＡＶ機器にＧＴに加えて、ソフトウェアを追加することにより個人認証、オンライン課金機能などをＡＶ機器に与える事も可能である。これらの機能も、ハードウェア TRM と同程度の強度で実現することができる。

続いて、第１０実施形態として、移動エージェント保護への応用について説明する。

ＧＴは、エージェントが移動先で、不正に耐えて使命を全うするための保護

機能を実現することができる。つまり、ＧＴは、耐攻撃エージェントとして、VHTRM に対して安全な移動機能を提供することができる。移動エージェントを耐攻撃化することにより、第１１実施形態において説明する GRID 計算への GT 適用時と同様のセキュリティ機能を実現することができる。

続いて、第１１実施形態としてＧＲＩＤ計算への応用について説明する。

ＧＲＩＤ計算やネットワークエージェントでは、従来、以下の問題があった。

１．完全性(Integrity)：GRID 計算の依頼先において、依頼した実行コードが正しい CPU で、正しいデータを用いて、正しく実行されているかどうかを確認することはできなかった。そのため、依頼先ユーザがどのような不正を働いたとしても、また依頼先の計算機に実行コードを書き換えるウィルスが入ったとしても、確実に確認する手段がなかった。

２．秘匿(Confidentiality)：計算の依頼先において、コードやデータの漏洩や不正コピーが発生する怖れがあった。

３．課金(Accounting)：計算の依頼先において、課金処理の不正や CPU 利用量データの改ざんが行われる怖れがあったため、課金処理や課金データの完全性を保証する必要があった。

４．上記の問題を解決するためにソフトウェアＴＲＭを用いると、処理が格段に遅くなってしまうため、処理速度を重視する GRID 計算の要件には見合わなくなってしまっていた。しかも、計算機に関する特定の知識があればソフトウェアＴＲＭを容易に破ることができた。

このような問題があるため、GRID 計算の依頼先として、一般のパーソナルコンピュータやワークステーションを積極的に利用することはできなかった。

しかし、ＧＴを実装したＣＰＵ上で稼動する保護コードとして、計算依頼先で実行されるソフトウェアを開発することにより、以下が実現されるため、上記問題が解決される。

１．安全なＣＰＵ（ＧＴ）の認証と実行コード改ざん防止保護

２．計算処理改ざん防止機能

３．安全な課金

４．実行コードの不正コピー、不正利用の防止

５．計算依頼先選択の最適化

６．信頼の必要なものは TRM のレベルと証明書の発効日などを指定できるようにする。

第１２実施形態として、ロボットへの応用について説明する。

人間や動物の作業や動作を肩代わりする自律型ロボットの研究やその発表が盛んであるが、その安全性についての検討もさらに重要になってくる。これまでは単なる情報処理装置としての計算機がウィルスなどにのっとられる脅威であったが、物理的に移動し、その用途によっては人間の力をも凌駕するロボットが同様の事態になることを想定する必要がある。しかし、ロボットに以下の機構を備える事により、このような問題を解決することが可能となる。

１．ロボット用の危険な部品の CPU にすべて GT を実装し、ロボット認定機関が発行した証明書が入っていることとする。

２．ロボット用の CPU は署名が確認できたコードしか実行しない機構を持つ。

３．ロボット用の CPU は証明書のない CPU とはセキュリティレベルの高い情報交換をしないこととする。

４．「殺人・傷害禁止ルール」を GT 保護ソフトウェアとして実現し、認定機関が発行した証明書の秘密鍵を埋め込む。

５．「殺人・傷害禁止ルール」実現ソフトウェアをルールに従って利用したアプリケーションのみに認定機関が発行した証明書の秘密鍵を埋め込む。

６．ロボットの全入力処理と危害を与えうる動作処理のプログラムコードすべてから、このルールエンジンを利用するようにし、全体を GT 保護化する。

７．プログラムコードにはかならずデジタル署名を付加して置くようにする。コードを実行する前に、署名チェックを行うようにし、実行許諾のない場合は停止する。

８．統合ソフトウェア全体をGT保護化する。

これにより、ロボットを凶悪犯化するウィルスや中央制御機能改ざんなどに対抗することが可能となる。

次に、第１３実施形態として、個人情報保護への応用について説明する。

今日、「.NET」のような万人からの信頼を獲得した個人情報管理サービスが多くの個人情報を管理しており、他のサービスは、自サービスを提供するために必要な個人情報およびそれらの統計情報程度しか得ることができない。従って、これらのサービスは、営業戦略上の顧客統計情報を獲得したり、広告メールサービスを提供したりするためには、特定の個人情報管理サービスを利用する必要がある。このような状況におけるセキュリティ上の課題は次のようなものである。

１．個人情報を回収するサービスが、サービス利用者に提示した「個人情報保護ポリシー」に従わない可能性がある。P3P（Platform for Privacy Preference：プライバシー制御のための基盤）を用いるなどにより、ポリシーを交換できたとしてもこれを強制する技術はない。

２．個人情報を扱うサービスにおいて、個人情報を処理する従業員自身が不正な個人情報漏洩を行う可能性がある。

これら問題を解決するために、本実施形態によれば、各サービスを提供するサーバの CPU にＧＴを実装し、その上でサーバに、個人情報を扱うサーバ DRM ソフトウェアやサーバアプリケーションをＧＴ保護コードとしてパッケージ化する。上述のように、ＧＴライセンスには、ソフトウェアを実行するプロセスに対してアクセス条件を設定する事ができる。従って、これまで、アクセス制

御を外部から強制する事ができなかったサーバに対しても、アプリケーション操作におけるアクセス条件を強制する事ができるようになる。

これにより、一般消費者に対して、信頼できる個人情報管理サービスを提供することができるようになる。さらに、サービスサイトへの個人情報の積極的な提供を促し、利用者の要求に従ったサイト間の積極的な個人情報交換を可能とする事により、広告・営業活動と消費活動をより活性化することが可能となる。

次に、第１４実施形態として、ウィルス対抗手段への応用について説明する。

従来のウィルス対抗手段では、ソフトウェアのデジタル署名チェック機能とウィルスチェックソフトウェアを併用している。しかしこの方式では、最新のウィルスが、署名チェック機能またはウィルスチェックソフトが起動するまでに動作する実行ファイルを書き換えることによる攻撃に対しては無力である。

コード署名逐次確認機能を持つＧＴであれば、このようなウィルスに対抗できる。そのためには、ＣＰＵに第２実施形態に係わるＧＴを搭載し、実装コードチェックのためのソフトウェア（コードとデータ）をＧＴで保護する。そして、ブートからウィルスチェックソフトウェアが起動されるまでコード署名をＧＴ（ＣＰＵ）が逐次確認する。その上で、ウィルスチェックソフトウェアをＧＴの耐攻撃モードで実行する。ウィルスチェックソフトウェアは、各コードを実行する前にコードのハッシュを計算したり、著名を確認したりする。

これにより、新種ウィルスに対抗できなかったシステム上のセキュリティホールを壊滅することができる。なお、コード署名チェック機能については、既に説明した。

このように、ＧＴをウィルス対抗手段に採用することにより、従来の方式に比較して格段に強度の高いウィルス対抗ソフトを実現することができる。

以上、ＧＴの様々な応用例について説明した。このように、ＧＴを搭載した

CPU を採用することにより、情報セキュリティ上の脅威により抑制されてきた各種先端的情報技術の社会基盤への適用が、促進されることが期待できる。例えば、次のような産業上の技術革新が期待できる。

1．コンテンツ保護の強度的及び機能的問題で積極的に PC に提供されなかったコンテンツが積極的にインターネットに流され、また各種 P2P (Person to Person)技術を用いて積極的に超流通が促進される。

2．利用者認証、電子財布などへの応用により、一般の情報機器のみを利用した買い物がソフトウェアの追加だけで安全にできる。これにより、インターネットでのお買い物を恐れていた利用者も買い物をするようになる。売る側も安心してサイトに品物を置いて販売できるようになり、市場がひろがり、グローバルなインターネット販売ビジネスを促進する。

3．強力な個人情報保護機能の実現により、個人情報を安心して積極的に売り込むことによるインターネットを介した広告活動と消費活動のさらなる活性化を期待できる。

4．見ず知らずの CPU の利用を恐れたり、無償で CPU を貸し出す不公平感により積極的に計算パワーを提供できなかったりしていた GRID 計算を積極的に活用できるようになる。これにより、一般の共用 CPU の利用効率が 10 倍程度以上には向上すると考えられる。従って、単純に計算すれば、各コンピュータの処理速度が、ローカルな CPU だけで計算していた時に比較し、平均して 10 倍程度以上の速さが期待できる。あるいは、必要な計算に世界の CPU 利用を集中できるようになる。

5．ソフトウェア部品の「超流通」とネットワークを越えた自律協調型分散開発を目標とする「超流用(Superappropriation)」を強力なウィルス対抗とモジュール利用料 P2P 課金のセキュリティで促進する。

6．GT による安全性の保障が、強力な実用ロボットの社会への浸透を積極的

に促進させる。

さらに将来、GT はいつでもどこでもだれにでも利用できる安全な汎用計算処理基盤として、高信頼ユービキタスコンピューティング（Trusted ubiquitous computing）のインフラにもなることができる。

以上、本発明の実施形態について説明したが、本発明は上述した実施形態に限定されるものではなく、他の様々な変更が可能である。

請求の範囲

1．　プログラムを実行する中央演算装置であって、

ブロックの暗号化及び暗号化ブロックの復号を行う暗号手段を備え、

前記中央演算装置には、第１の秘密鍵が秘密裏に隠され、

前記暗号手段は、前記第１の秘密鍵と対になった公開鍵を用いて暗号化された第１のプログラムの第１のライセンスを、前記第１の秘密鍵を用いて復号することにより、前記第１のライセンスから前記第１のプログラムを構成する暗号化ブロックを復号するためのコード復号鍵を取得する、

ことを特徴とする中央演算装置。

2．　キャッシュを更に備え、

前記暗号手段は、前記第１のプログラムを構成する暗号化ブロックがメモリ領域から前記キャッシュに出力される際に、キャッシュ単位で前記暗号化ブロックを復号する、

ことを特徴とする請求項１に記載の中央演算装置。

3．　ユーザが参照したり改ざんしたりすることができない耐攻撃バッファを更に備え、

前記コード復号鍵は、前記耐攻撃バッファに記録される、

ことを特徴とする請求項１又は２に記載の中央演算装置。

4．　前記第１のライセンスは、前記第１のプログラムの実行プロセスがメモリ領域にアクセスする際のアクセス条件を含み、

前記中央演算装置は、前記第１のプログラムを構成する暗号化ブロックが記録される前記メモリ領域のアドレスと前記メモリ領域へのアクセス条件とを記録するＴＬＢ（Translation Look aside Buffer）と、

メモリ管理手段と、

キャッシュと、

プロセッサコアを更に備え、

前記ＴＬＢと前記耐攻撃バッファはリンクされ、

前記メモリ管理手段は、暗号化ブロックを記録するメモリ領域のアドレスに基づいて前記ＴＬＢから前記メモリ領域へのアクセス条件を取得し、さらに、前記耐攻撃バッファから、前記メモリ領域に対応する前記コード復号鍵を取得し、

前記プロセッサコアは、前記メモリ管理手段が取得した前記アクセス条件に基づいて、前記実行プロセスから前記メモリ領域へのアクセスを実行することが許可されているか否か判定し、前記メモリ領域へのアクセスを実行する事が許可されていると判定した場合、前記実行プロセスから前記メモリ領域へのアクセスを実行し、

前記暗号手段は、前記メモリ管理手段が取得した前記コード復号鍵を用いて前記メモリ領域内の前記暗号化ブロックを復号して得られたコードを前記キャッシュに書き込む、

ことを特徴とする請求項３に記載の中央演算装置。

５．　前記コード復号鍵と、前記暗号化ブロックを暗号化する際に用いられた暗号鍵は同一の鍵である、

ことを特徴とする請求項４に記載の中央演算装置。

６．　前記第１のプログラムの実行プロセスからアクセスされるメモリ領域が、第１のメモリ領域から第２のメモリ領域に切り替わった場合、

前記メモリ管理手段は、さらに、前記耐攻撃バッファから取得された前記第１のメモリ領域に対応するコード復号鍵と、前記第２のメモリ領域に対応するコード復号鍵とが一致するか否か判定し、一致すると判定した場合、前記実行プロセスから前記第２のメモリ領域へのアクセスを実行し、一致しないと判定

した場合、前記実行プロセスから前記第２のメモリ領域へのアクセスを実行しない、

ことを特徴とする請求項４又は５に記載の中央演算装置。

７．　前記第１のライセンスは、前記第１のプログラムに埋め込まれている、

ことを特徴とする請求項１乃至６のいずれかに記載の中央演算装置。

８．　前記耐攻撃バッファには前記コード復号鍵ごとに、異なるデータ暗号鍵が記録されており、

前記暗号手段は、前記キャッシュ内のデータを前記メモリ領域に記録する際には、前記データを前記データ暗号鍵を用いて暗号化した後、前記ＴＬＢによって前記データ暗号鍵に対応付けられた前記メモリ領域に記録し、前記メモリ領域内の暗号化されたデータを読み出す際には、読み出された前記データを前記データ暗号鍵を用いて復号した後、前記キャッシュに書き込む、

ことを特徴とする請求項４乃至７のいずれかに記載の中央演算装置。

９．　第１のコードを実行することにより得られたデータを第２のコードで利用する場合、前記プロセッサコアは、前記データを記録するメモリ領域へのアクセス権を前記第２のコードに与えるように前記ＴＬＢを設定し、且つ、前記データを暗号化するためのデータ暗号鍵を、前記第２のコードが前記データを前記メモリ領域から読み出す際に使用するように前記ＴＬＢ及び前記耐攻撃バッファを設定する、

ことを特徴とする請求項８に記載の中央演算装置。

１０．　レジスタと、

レジスタへのアクセス制御を行うためのレジスタアクセス制御テーブルと、を更に備え、

前記プロセッサコアは、前記レジスタアクセス制御テーブル内の封印フラグを用いて、前記レジスタの封印及び解放を制御する、

ことを特徴とする請求項４乃至９のいずれかに記載の中央演算装置。

１１．　ＴＬＢの内容を外部記憶装置内のページテーブルに記録する際には、前記暗号手段は、記録する内容に署名を付与し、

前記ページテーブルの内容を前記ＴＬＢに取り込む前には、前記暗号手段は、前記署名が正しいことを確認する、

ことを特徴とする請求項４乃至１０のいずれかに記載の中央演算装置。

１２．　前記耐攻撃バッファの内容を外部記憶装置内の暗号化キーテーブルに記録する際には、前記暗号手段は、記録する内容を暗号化する、

ことを特徴とする請求項３乃至１１のいずれかに記載の中央演算装置。

１３．　前記中央演算装置は、他の中央演算装置と接続され、

前記中央演算装置は、前記他の中央演算装置と相互に認証を行う事にことによりセッション鍵を取得し、前記中央演算装置の前記暗号手段は、前記セッション鍵を用いて前記中央演算装置の前記キャッシュの内容を暗号化して、前記他の中央演算装置に同期転送する、

ことを特徴とする請求項１乃至１２のいずれかに記載の中央演算装置。

１４．　前記暗号手段は、前記第１のプログラムを実行する前に、第２のプログラムに付加された前記第２のライセンスを前記公開鍵を用いて復号する事により第２の秘密鍵を暗号化する際に用いられた秘密鍵暗号鍵を取得し、さらに、前記取得された秘密鍵暗号鍵を用いて前記第２の秘密鍵を復号する、

ことを特徴とする請求項１乃至請求項１３のいずれかに記載の中央演算装置。

１５．　前記第２のライセンスには、前記第１のプログラムの実行プロセスから読み出しのみ可であることを示すアクセス条件が付加されており、

前記第２の秘密鍵は、前記第１のプログラムの実行プロセスからの読み出しのみ可能である、

ことを特徴とする請求項１４に記載の中央演算装置。

１６．　前記第２の秘密鍵は、データ暗号鍵によって暗号化されてメモリ領域に記録される、

ことを特徴とする請求項１４又は１５に記載の中央演算装置。

１７．　前記耐攻撃バッファは、さらに、

対応する前記耐攻撃バッファ内の情報を前記耐攻撃バッファ外に出力しても良いか否かを示す外部出力禁止情報と、

対応する情報をキャッシュ外に出力しても良いか否かを示すキャッシュロック情報とを記録し、

前記外部出力禁止情報及び前記キャッシュロック情報に基づいて、第１のプログラム及び他のプログラムとの間における前記第１のライセンスの移動は管理される、

ことを特徴とする請求項３乃至１６のいずれかに記載の中央演算装置。

１８．　前第１のプログラムは、トラステッドコンピューティングモジュールである、

ことを特徴とする請求項１乃至１７のいずれかに記載の中央演算装置。

１９．　前記第１のプログラムは、前記中央演算装置に電子財布を実現させるプログラムである、

ことを特徴とする請求項１乃至１７のいずれかに記載の中央演算装置。

２０．　前記第１のプログラムは、個人情報を扱うプログラムである、

ことを特徴とする請求項１乃至１７のいずれかに記載の中央演算装置。

２１．　前記第１のプログラムは、前記中央演算装置の実装コードのウィルスチェックプログラムである、

ことを特徴とする請求項１乃至１７のいずれかに記載の中央演算装置。

２２．　前記第１のプログラムは複数の中央演算装置間を移動する移動エージェントである、

ことを特徴とする請求項１乃至１７のいずれかに記載の中央演算装置。

２３．　前記第１のプログラムを構成するブロックは、前記ブロックのハッシュ値の確認が必要であるか否かを示すハッシュ確認要否情報を含み、

前記ハッシュ確認要否情報に基づいて、前記ブロックのハッシュ値を算出し、前記ブロックに付加するハッシュ手段と、

前記ハッシュ確認要否情報に基づいて、前記ブロックの前記ハッシュ値を確認するハッシュ確認手段と、

を更に備えることを特徴とする請求項１乃至２２のいずれかに記載の中央演算装置。

２４．　前記第１のプログラムを構成するブロックは、前記ブロックが保護を必要とするか否かを示す暗号化要否情報を含み、

前記暗号化要否情報に基づいて、前記ブロックを前記暗号手段に出力するか、そのままキャッシュ又はメモリ領域に出力するか判定する保護ブロック選択手段を、

更に備えることを特徴とする請求項１乃至２３のいずれかに記載の中央演算装置。

２５．　前記第１のプログラムの実行ファイルのヘッダは、前記第１のプログラムを構成するブロックの構成を示す暗号化ブロックビットマップを含み、

前記暗号化ブロックビットマップに基づいて、前記ブロックを前記暗号手段に出力するか、そのままキャッシュ又はメモリ領域に出力するか判定する保護ブロック選択手段を、

更に含むことを特徴とする請求項１乃至２３のいずれかに記載の中央演算装置。

２６．　前記第１のプログラムのコードの先頭は、前記第１のプログラムを構成する複数のブロックが、平文ブロックと暗号化ブロックの組合せの繰り返し

であることを指定し、且つ、前記組合せにおいて平文ブロックが連続する数及び暗号化ブロックが連続する数を指定するコードであり、

前記コードを実行することにより、前記プロセッサコアは、前記ブロックを前記暗号手段に出力するか、そのままキャッシュ又はメモリ領域に出力するか判定する、

更に含むことを特徴とする請求項１乃至２３のいずれかに記載の中央演算装置。

２７．　前記キャッシュとメモリの間に、

前記暗号手段を介するキャッシュラインと、

前記暗号手段を介さないキャッシュラインと、

を更に備えることを特徴とする請求項２乃至２６のいずれかに記載の中央演算装置。

２８．　請求項１乃至２７のいずれかに記載の中央演算装置を有することを特徴とするコンピュータ。

２９．　請求項１乃至２７のいずれかに記載の中央演算装置を有することを特徴とするＩＣカード。

３０．　前記第１のプログラムは、前記ＩＣカードのセキュリティ機能を実現するプログラムである、

ことを特徴とする請求項２９に記載のＩＣカード。

３１．　前記中央演算装置は、ロボットに搭載され、

前記第１のプログラムは、前記ロボットを制御する制御プログラムである、

ことを特徴とする請求項１乃至２７のいずれかに記載の中央演算装置。

３２．　中央演算装置に保護プログラムを実行する許諾を与える制御を行わせるプログラムであって、

前記保護プログラムは、コード暗号鍵によって暗号化され、

前記保護プログラムに対応して、前記コード暗号鍵を含み、且つ、前記中央演算装置に秘密裏に備えられた秘密鍵と対になる公開鍵によって暗号化されたライセンスが存在し、

前記中央演算装置が前記保護プログラムを実行する前に、前記ライセンスを前記中央演算装置に投入し、

前記中央演算装置に備えられた暗号手段に、前記秘密鍵を用いて前記ライセンスを復号することにより、前記ライセンスから前記コード暗号鍵を取得させ、

前記暗号手段に、前記コード暗号鍵を用いて前記保護プログラムを復号させる、

ことを含む処理を前記中央演算処理装置に実行させるプログラム。

３３．　中央演算装置に、保護プログラムを実行する許諾を与える制御を行わせるプログラムを記録する記録装置であって、

前記保護プログラムは、コード暗号鍵によって暗号化され、

前記保護プログラムに対応して、前記コード暗号鍵を含み、且つ、前記中央演算装置に秘密裏に備えられた秘密鍵と対になる公開鍵によって暗号化されたライセンスが存在し、

前記中央演算装置が前記保護プログラムを実行する前に、前記ライセンスを前記中央演算装置に投入し、

前記中央演算装置に備えら得た暗号手段に、前記秘密鍵を用いて前記ライセンスを復号することにより、前記ライセンスから前記コード暗号鍵を取得させ、

前記暗号手段に、前記コード暗号鍵を用いて前記保護プログラムを復号させる、

ことを含む処理を前記中央演算装置に実行させるプログラムを記録する記録装置。

３４．　中央演算装置に、保護プログラムを実行する許諾を与えるプログラム

実行許諾方法であって、

前記保護プログラムは、コード暗号鍵によって暗号化され、

前記保護プログラムに対応して、前記コード暗号鍵を含み、且つ、前記中央演算装置に備えられた秘密鍵と対になる公開鍵によって暗号化されたライセンスが存在し、

前記中央演算装置は、前記保護プログラムを実行する前に、前記ライセンスを取得し、

前記中央演算装置は、前記秘密鍵を用いて前記ライセンスを復号することにより、前記ライセンスから前記コード暗号鍵を取得し、

前記中央演算装置は、前記コード暗号鍵を用いて前記保護プログラムを復号する、

ことを含むことを特徴とするプログラム実行許諾方法。

３５．　コンピュータにおいて実行されるプログラムコードであって、

前記プログラムコードは、コード暗号鍵によって暗号化され、

前記プログラムコードに対応して、前記コード暗号鍵を含み、且つ、前記プログラムコードを実行するべきコンピュータに備えられた中央演算装置が秘密裏に備える秘密鍵と対になる公開鍵によって暗号化されたライセンスが存在し、

前記ライセンスは前記プログラムコードが実行される前に前記中央演算装置に投入され、

前記中央演算装置によって前記秘密鍵を用いて前記ライセンスが復号され、

前記プログラムコードは、前記ライセンスから取得された前記コード暗号鍵を用いて、前記中央演算装置によって復号される、

ことを特徴とするプログラム。

３６．　コンピュータにおいて実行されるプログラムコードを記録する、前記コンピュータによって読み取り可能な記録媒体であって、

前記プログラムコードは、コード暗号鍵によって暗号化され、

前記プログラムコードに対応して、前記コード暗号鍵を含み、且つ、前記プログラムコードを実行するべきコンピュータに備えられた中央演算装置が秘密裏に備える秘密鍵と対になる公開鍵によって暗号化されたライセンスが存在し、

前記ライセンスは前記プログラムコードが実行される前に前記中央演算装置に投入され、

前記中央演算装置によって前記秘密鍵を用いて前記ライセンスが復号され、

前記プログラムコードは、前記ライセンスから取得された前記コード暗号鍵を用いて、前記中央演算装置によって復号される、

ことを特徴とするプログラムを記録する記録媒体。

３７．　秘密裏に隠された秘密鍵と、暗号化及び復号を行う暗号手段とを備える中央演算装置を備えるコンピュータにおいて実行されるプログラムを生成するプログラム生成装置であって、

コードオブジェクトを入力する入力手段と、

入力された前記コードオブジェクトを複数のブロックに分割し、それぞれのブロックにＮＯＰ命令を追加するリンカ前処理手段と、

アドレス解決を行うリンカ手段と、

各ブロックをコード暗号鍵を用いて暗号化することにより保護コード実行形式を生成する保護コード実行形式生成手段と、

前記コード暗号鍵を含み、前記秘密鍵と対になる公開鍵によって暗号化されたライセンスを生成するライセンス生成手段とを備え、

前記ライセンスは、前記コンピュータが前記保護コード実行形式を実行する前に前記中央演算装置に投入され、前記暗号手段によって前記秘密鍵を用いて復号され、

前記保護コード実行形式は、前記ライセンスから取得された前記コード暗号

鍵を用いて、前記暗号手段によって復号される、

ことを特徴とするソフトウェア生成装置。

要約書

中央演算装置（ＧＴ）１０は、秘密裏に秘密鍵と、暗号エンジン１６を備える。ＧＴ１０が、ソフトウェアを実行する前には、ソフトウェアに付加されたライセンスがＧＴ１０に投入される。ライセンスは、ソフトウェアを暗号化する際に用いられたコード暗号鍵を、秘密鍵と対になる公開鍵で暗号化した情報を含む。

ライセンスが投入されると、暗号エンジン１６は、秘密鍵を用いてライセンスを復号することによりコード暗号鍵を取得し、そのコード暗号鍵を用いてソフトウェアを復号する。

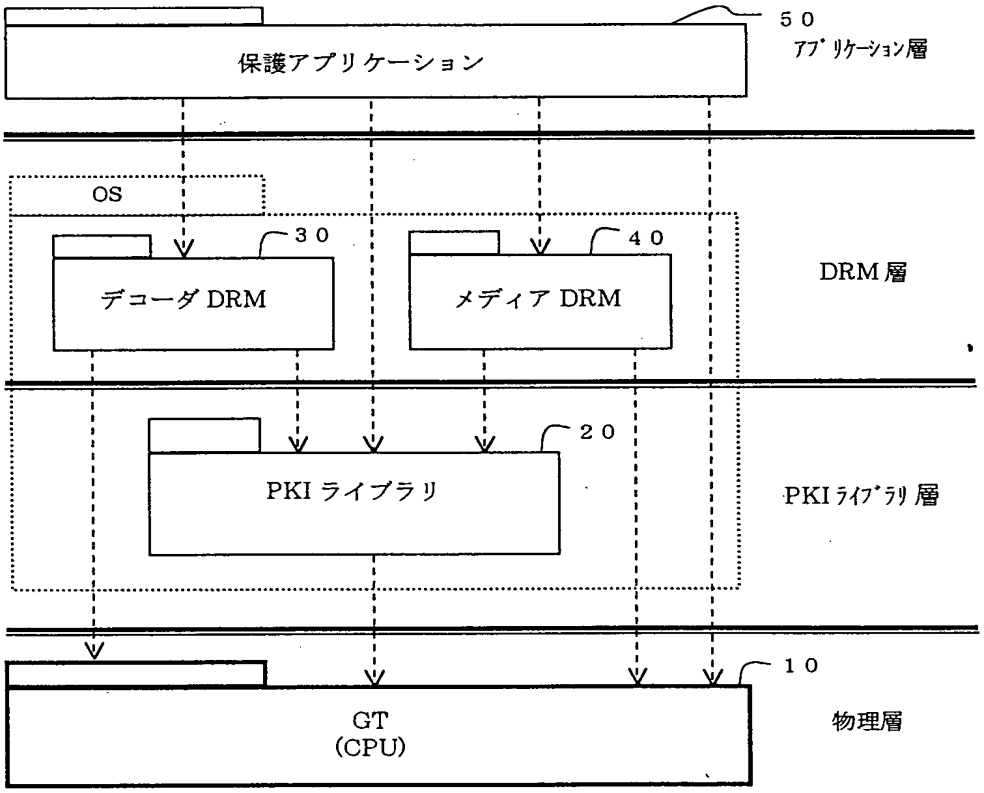
50
保護アプリケーション
アプリケーション層
OS
30
デコーダ DRM
40
メディア DRM
DRM 層
20
PKI ライブラリ
PKI ライブラリ 層
10
GT
(CPU)
物理層

図 1

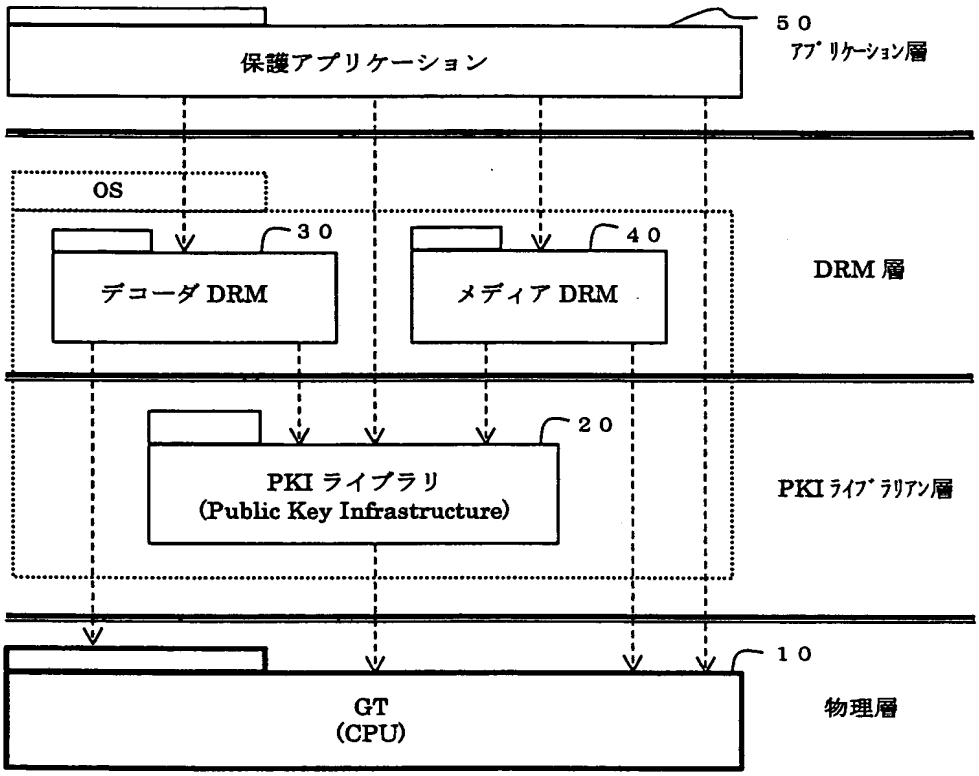
50
ｱﾌﾟﾘｹｰｼｮﾝ層
保護アプリケーション
OS
30
デコーダ DRM
40
メディア DRM
DRM 層
20
PKI ライブラリ
(Public Key Infrastructure)
PKI ﾗｲﾌﾞﾗﾘｱﾝ層
10
GT
(CPU)
物理層

図 1

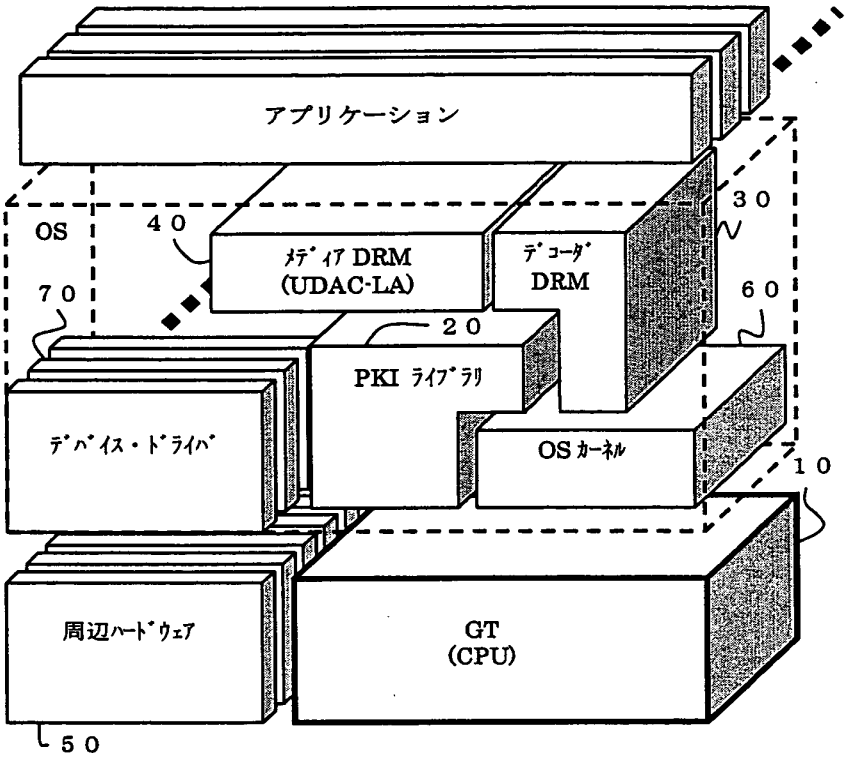
アプリケーション
OS
40
メディア DRM
(UDAC-LA)
デコーダ
DRM
30
70
60
20
PKI ライブラリ
デバイス・ドライバ
OS カーネル
10
周辺ハードウェア
GT
(CPU)
50

図2

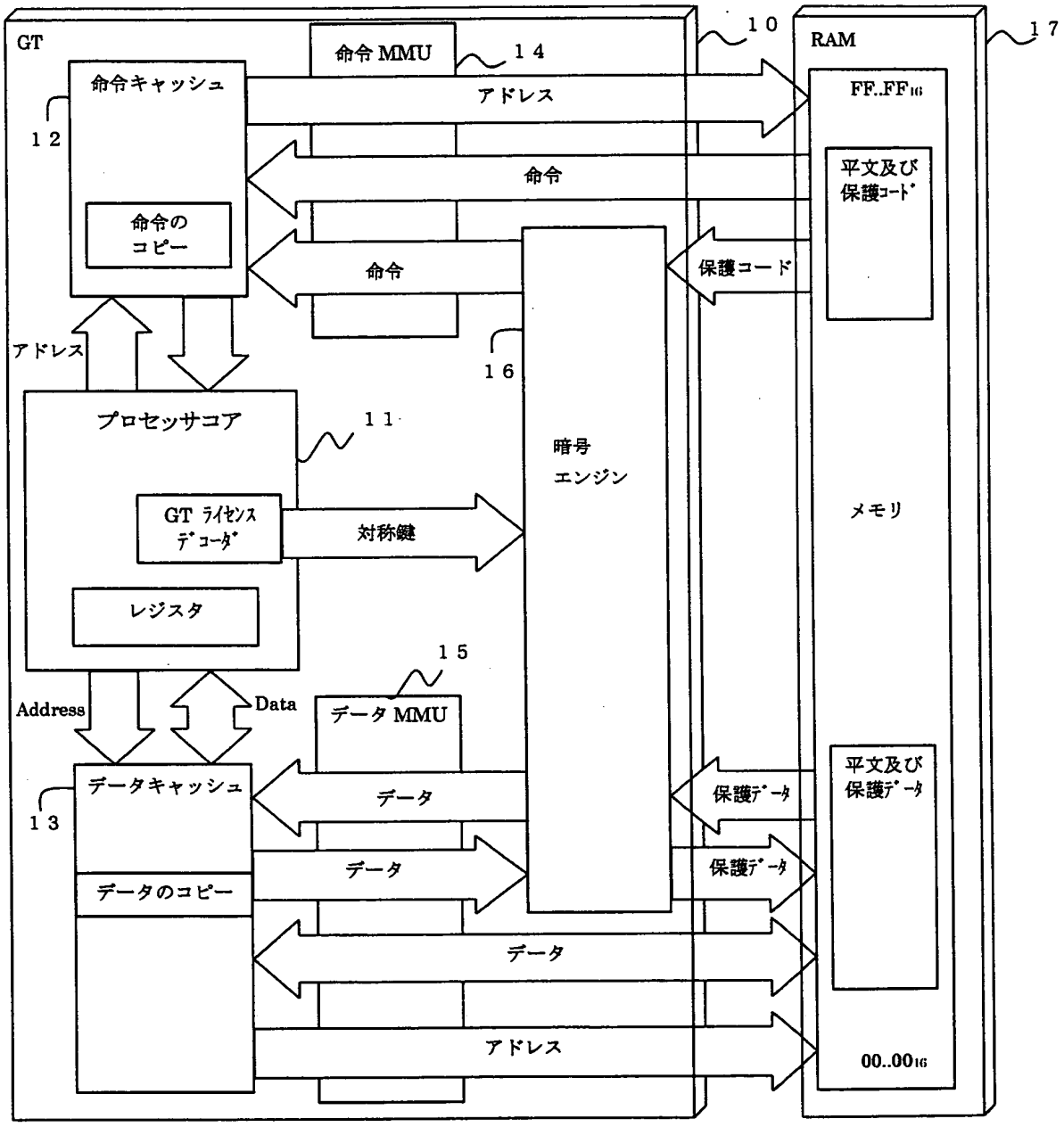
GT
命令 MMU
14
10
RAM
17
命令キャッシュ
12
アドレス
FF..FF16
命令
平文及び
保護ｺｰﾄﾞ
命令の
コピー
命令
保護コード
16
アドレス
プロセッサコア
11
暗号
エンジン
GT ﾗｲｾﾝｽ
ﾃﾞｺｰﾀﾞ
対称鍵
メモリ
レジスタ
15
Address
Data
データ MMU
データキャッシュ
13
データ
保護ﾃﾞｰﾀ
平文及び
保護ﾃﾞｰﾀ
データ
保護ﾃﾞｰﾀ
データのコピー
データ
アドレス
00..0016

図３

Notation	Field Size	Name	Description
p	1 bit	Present flag 有効性フラグ	on(1)であれば，TRBが有効であることを示す。
uo	1 bit	Unable to Output 外部出力禁止フラグ	on(1)であれば，暗号化キーテーブルに出力されないことを示す。
cl	1 bit	Cache Lock キャッシュロックフラグ	on(1)であれば，kidが指定された耐攻撃ページのデータがキャッシュの外に出力されないことを示す。
kid	20 bit程度	Key ID 鍵識別情報	TLBからのリンクのための情報。
key	128 bit以上	Cryptographic Key 暗号鍵	このラインにリンクするページのコードまたはデータの暗号・復号鍵の値。対称鍵（共通鍵）暗号法の鍵
ackey	keyに同じ	Authorized code key 被許諾コード鍵	kidを含む全TLB行のページにアクセスが許可された保護プロセスの実行コード復号鍵(他行又は自行のkey)。
eadr	64 bit程度	Exception address 例外処理アドレス	keyが異なるページから，このTRBにリンクしたページに復帰する直前に，発生する例外処理コードの先頭アドレス

図４

Notation		Field Size	Name	Description
p		1 bit	Present flag 有効性フラグ	TLBが有効であることを示す。
id		20 bit程度	Region Identifier 領域識別子	TLB内の当該行が示すページ領域の識別子値。
ppn		32 bit程度	Physical page number 物理ページ番号	従来からのTLBの物理ページ番号。
vpn		56 bit程度	Virtual page number 仮想ページ番号	従来からのTLBの仮想ページ番号。
rights	pl	2 bit	Privilege level 権限レベル	ページにアクセス可能な権限のレベル。詳細は図６及び図７のとおり。
	ar	3 bit	Access Rights アクセス権	ページへのアクセス権の種類を指定する。詳細は図６及び図７のとおり。
	tr	1 bit	Tamper Resistance 耐攻撃性フラグ	on(1)であれば，耐攻撃セグメント空間にあり，TRBを持つことを示す。
	df	1 bit	Debug mode Flag デバッグモードフラグ	trがon(1)の場合のみ有効。trとdfがonならar指定ごとのデバッグモードで動作する。動作詳細は3．１．2を参照。
kid		20 bit程度	Key ID 鍵識別情報	TRB内の鍵の情報にリンクするためのTRB行(insertion)識別情報。
ebim		3 bit程度	Encrypted Block Identification 暗号化ブロック識別方式	GTライセンスのACgt.ebimの値のコピー。trがoff(0)のときにはこの値も0となる。
sign		128 bit程度	Digital signature デジタル署名	vpnからebimまでのフィールド連結データのデジタル署名値。GTが生成。

図５

PSR.S 実行モード	PTE.TR (TLB.tr)	PTE.PP (TLB.ar)	IF（命令フェッチ）	Load（ロード命令）	Store（ストア命令）
1 スーパーバイザモード	0 一般	000	OK	OK	OK
		001	OK	OK	NG
		010	OK	NG	NG
		011	NG	NG	NG
	1 耐攻撃	000	指定コード	指定コード	指定コード
		001	指定コード	指定コード	NG
		010	指定コード	NG	NG
		011	NG	NG	NG
0 ユーザモード	0 一般	100	OK	OK	OK
		101	OK	OK	NG
		110	OK	NG	NG
		111	NG	NG	NG
	1 耐攻撃	100	指定コード	指定コード	指定コード
		101	指定コード	指定コード	NG
		110	指定コード	NG	NG
		111	NG	NG	NG

図6

TLB.rights		TLB.pl	Process Privilege Level				Description
tr & df	ar		3	2	1	0	
tr=0	0	3	R	R	R	R	read のみ
		2	-	R	R	R	
or		1	-	-	R	R	
		0	-	-	-	R	
tr=1,	1	3	RX	RX	RX	RX	read, execute
		2	-	RX	RX	RX	
		1	-	-	RX	RX	
		0	-	-	-	RX	
	2	3	RW	RW	RW	RW	read, write
		2	-	RW	RW	RW	
		1	-	-	RW	RW	
		0	-	-	-	RW	
	3	3	RWX	RWX	RWX	RWX	read, write, execute
		2	-	RWX	RWX	RWX	
		1	-	-	RWX	RWX	
		0	-	-	-	RWX	
tr=1,	0	3	PR	PR	PR	PR	指定ﾌﾟﾛｾｽからのreadのみ
df=0		2	-	PR	PR	PR	
		1	-	-	PR	PR	
		0	-	-	-	PR	
	1	3	X	X	X	X	指定ﾌﾟﾛｾｽからの
		2	-	X	X	X	execute, read
		1	-	-	X	X	
		0	-	-	-	X	
	2	3	PRW	PRW	PRW	PRW	指定ﾌﾟﾛｾｽからの
		2	-	PRW	PRW	PRW	read, write
		1	-	-	PRW	PRW	
		0	-	-	-	PRW	
	3	3	PWX	PWX	PWX	PWX	指定ﾌﾟﾛｾｽからの
		2	-	PWX	PWX	PWX	read, write,execute
		1	-	-	PWX	PWX	
		0	-	-	-	PWX	

図 7

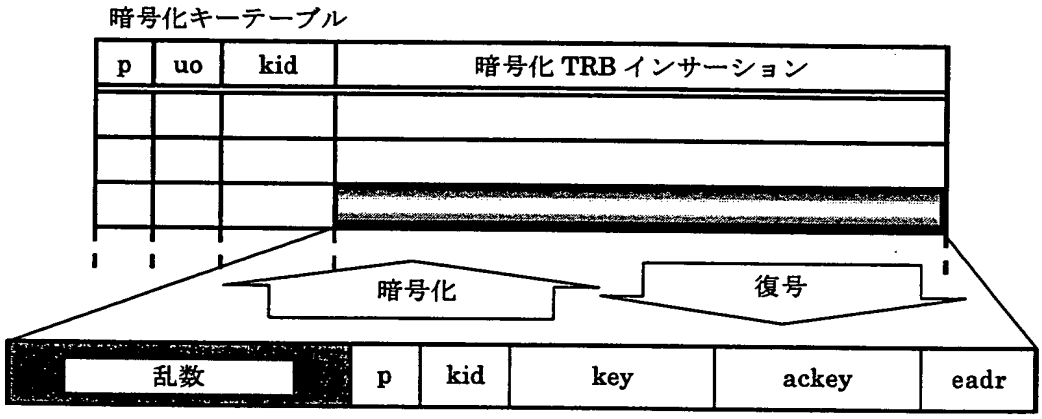
暗号化キーテーブル
p
uo
kid
暗号化 TRB インサーション
暗号化
復号
乱数
p
kid
key
ackey
eadr

図 8

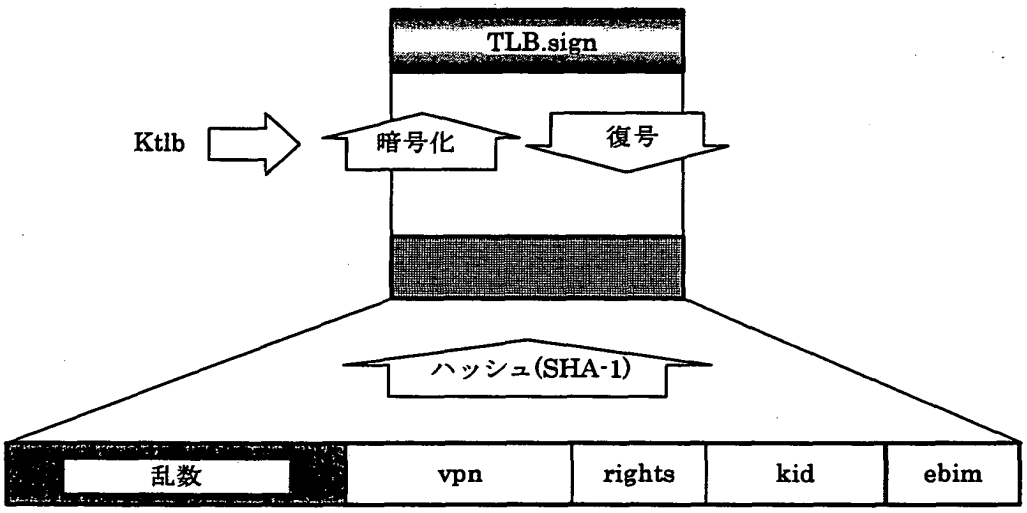

図9

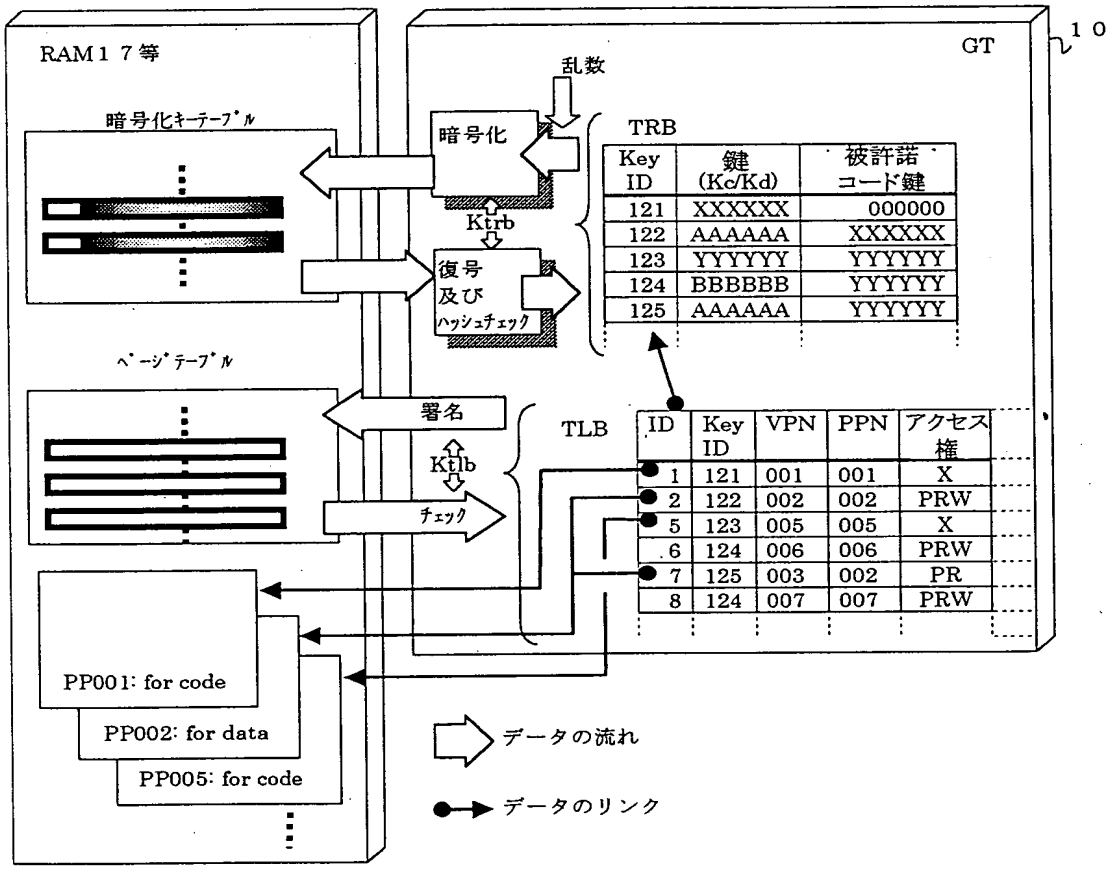
RAM17等
GT
10
乱数
暗号化キーテーブル
暗号化
Ktrb
復号
及び
ハッシュチェック
TRB
Key ID
鍵 (Kc/Kd)
被許諾 コード鍵
121 XXXXXX 000000
122 AAAAAA XXXXXX
123 YYYYYY YYYYYY
124 BBBBBB YYYYYY
125 AAAAAA YYYYYY
ページテーブル
署名
Ktlb
チェック
TLB
ID
Key ID
VPN
PPN
アクセス権
1 121 001 001 X
2 122 002 002 PRW
5 123 005 005 X
6 124 006 006 PRW
7 125 003 002 PR
8 124 007 007 PRW
PP001: for code
PP002: for data
PP005: for code
データの流れ
データのリンク

図１０

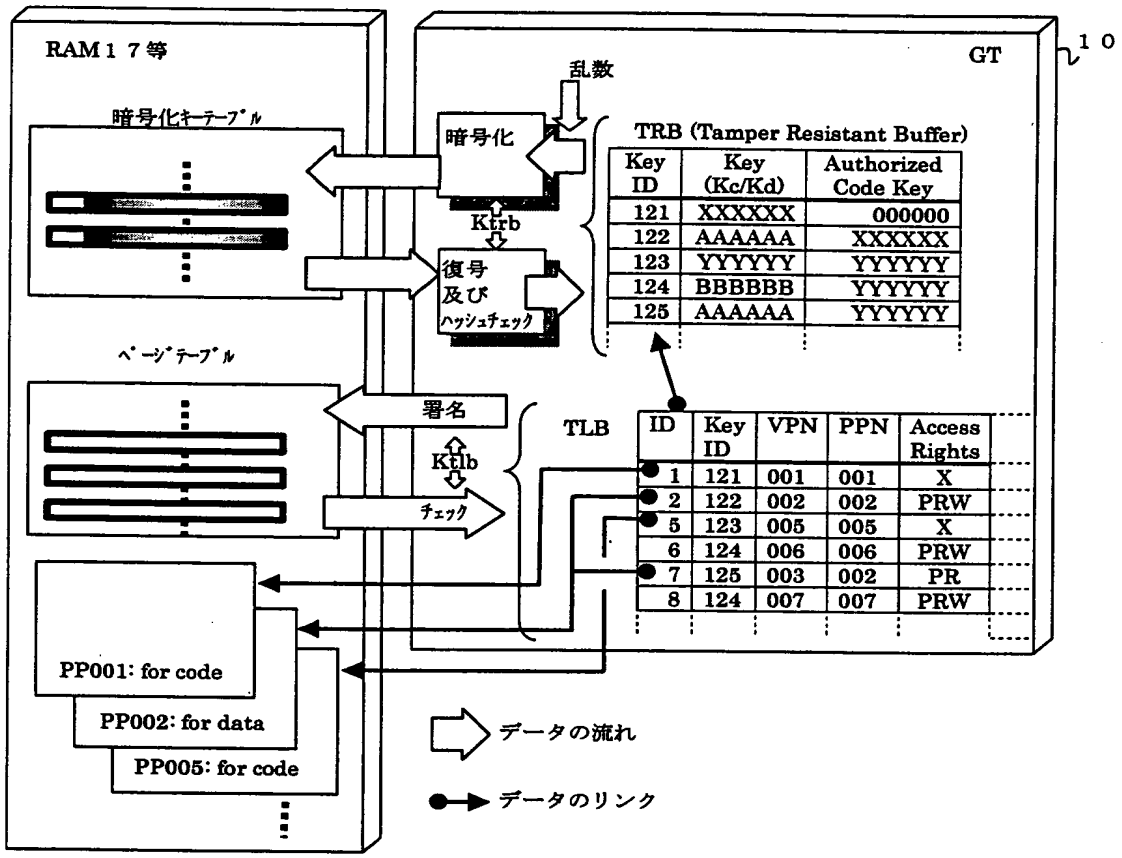
RAM１７等
暗号化ｷｰﾃｰﾌﾞﾙ
ﾍﾟｰｼﾞﾃｰﾌﾞﾙ
PP001: for code
PP002: for data
PP005: for code
GT
１０
乱数
暗号化
Krb
復号
及び
ﾊｯｼｭﾁｪｯｸ
署名
Ktlb
ﾁｪｯｸ
TRB (Tamper Resistant Buffer)
Key ID
Key (Kc/Kd)
Authorized Code Key
121 XXXXXX 000000
122 AAAAAA XXXXXX
123 YYYYYY YYYYYY
124 BBBBBB YYYYYY
125 AAAAAA YYYYYY
TLB
ID
Key ID
VPN
PPN
Access Rights
1 121 001 001 X
2 122 002 002 PRW
5 123 005 005 X
6 124 006 006 PRW
7 125 003 002 PR
8 124 007 007 PRW
データの流れ
データのリンク

図１０

Notation	Field Size	Name	Description
trss	1 bit	Tamper Resistant Segment Selector 耐攻撃セグメントセレクタ	onで耐攻撃空間が選択され，offで一般の仮想空間が選択される。

図１１

Notation	Field Size	Name	Description
rid	8 bit程度	Register ID	レジスタを指定するID
seal	1 bit	Seal flag 封印フラグ	on(1)で封印中であることを示し， off(0)で解放されていることを示す。
ackey	TRB.key に同じ	Authorized code key 被許諾コード鍵	レジスタへのアクセスが許可されて いるプロセスのコードページのKey

図１２

Notation	Field Size	Name	Description
r1ss	1 bit	Register 1 Seal Status レジスタ1封印状態	on(1)なら，r1が封印中であることを示し，off(0)なら解除されていることを示す。
r2ss	1 bit	Register 2 Seal Status レジスタ2封印状態	on(1)なら，r2が封印中であることを示し，off(0)なら解除されていることを示す。

図１３

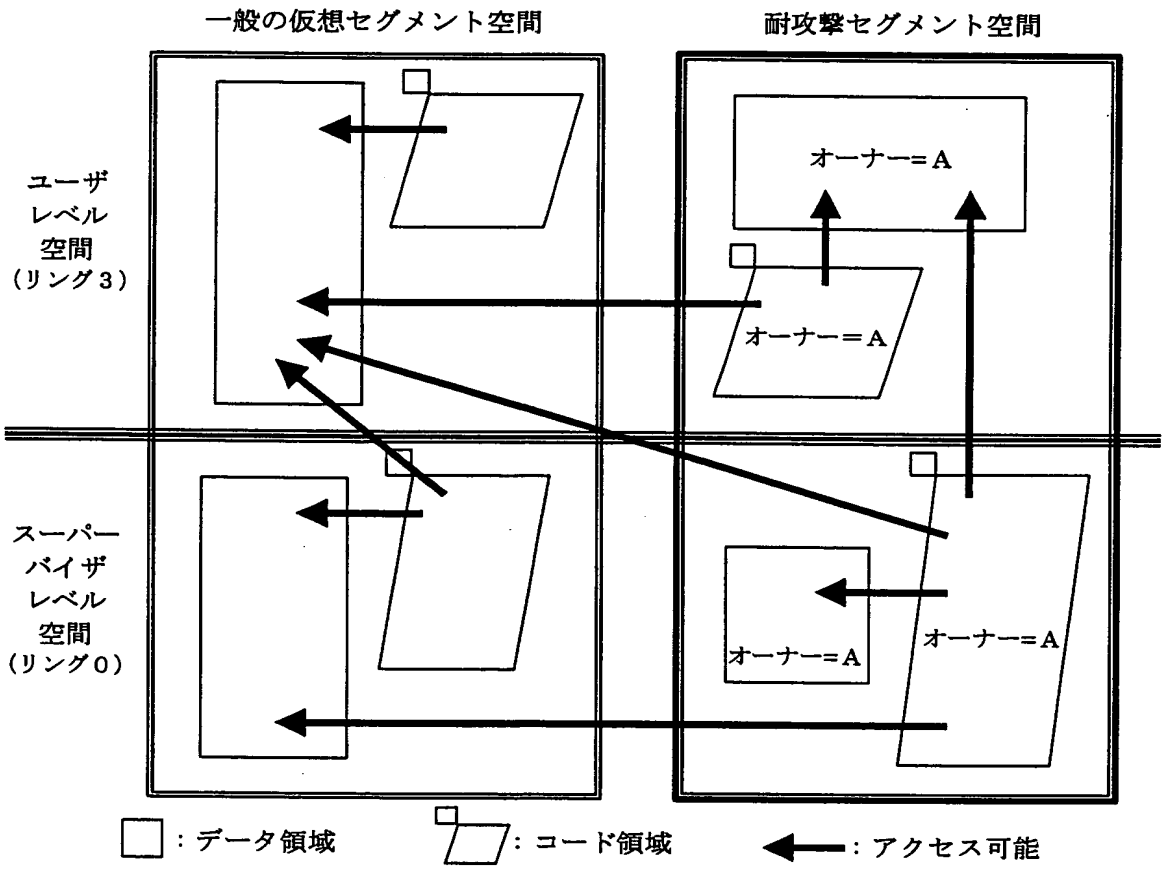
一般の仮想セグメント空間
耐攻撃セグメント空間
ユーザ
レベル
空間
（リング３）
スーパー
バイザ
レベル
空間
（リング０）
オーナー＝Ａ
オーナー＝Ａ
オーナー＝Ａ
オーナー＝Ａ
：データ領域
：コード領域
：アクセス可能

図１４

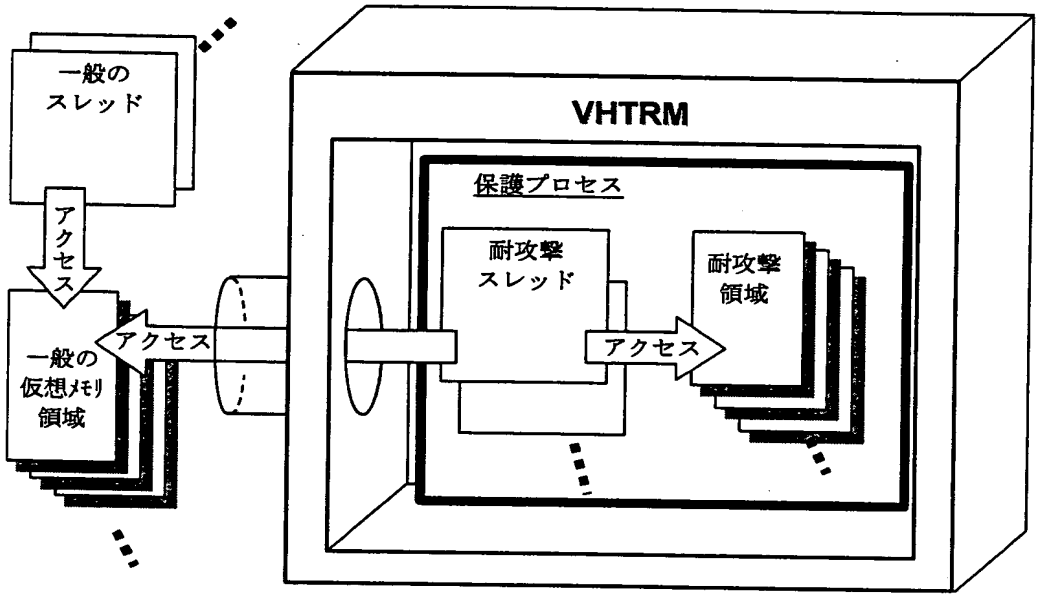
一般の
スレッド
アクセス
一般の
仮想メモリ
領域
アクセス
VHTRM
保護プロセス
耐攻撃
スレッド
アクセス
耐攻撃
領域

図１５

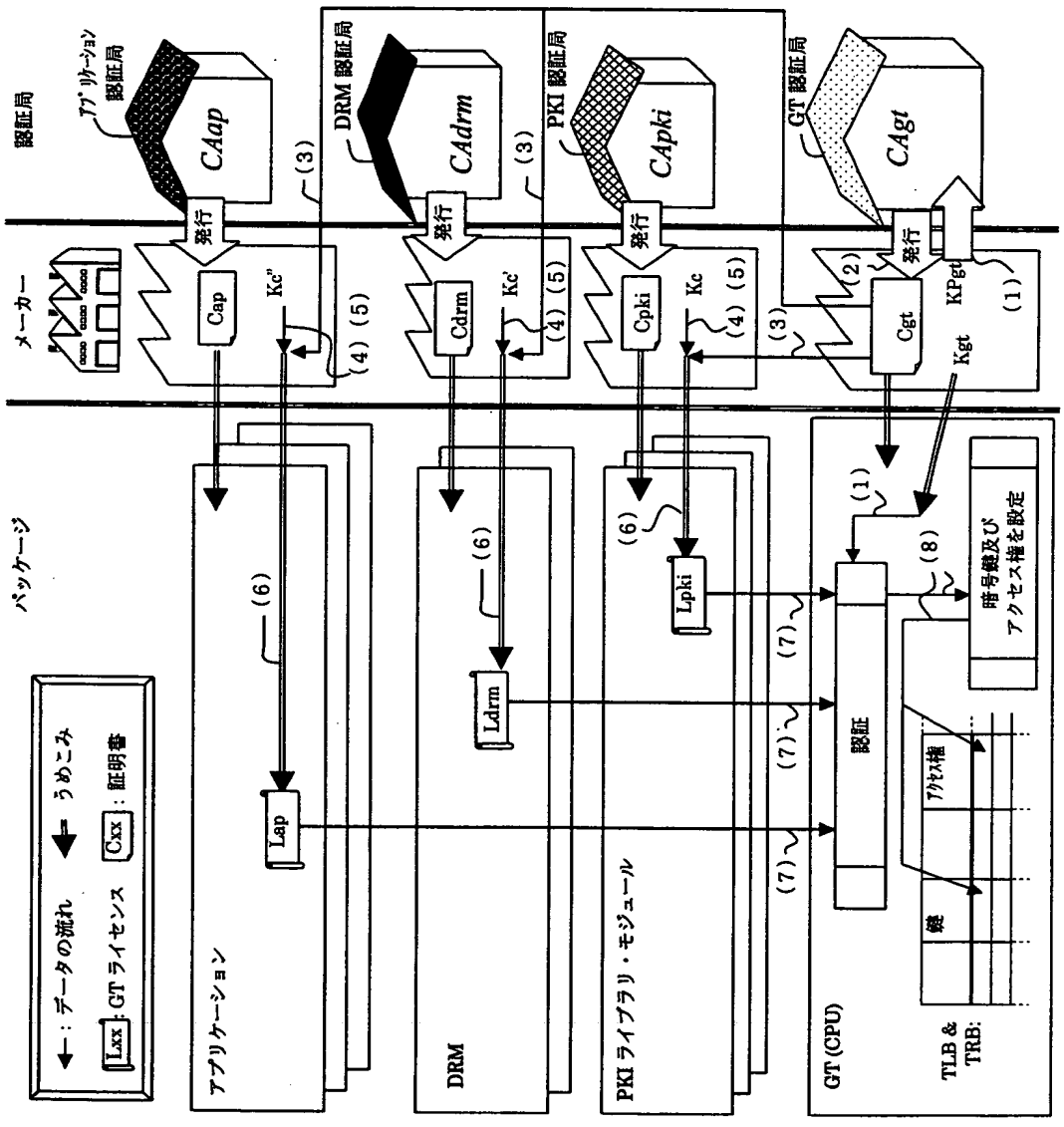
認証局
アプリケーション認証局
CAap
DRM 認証局
CAdrm
PKI 認証局
CApki
GT 認証局
CAgt
発行
メーカー
Cap
Kc'
(3)
(4) (5)
Cdrm
Kc'
Cpki
Kc
Cgt
KPgt
Kgt
(1)
(2)
パッケージ
(6)
Lap
Ldrm
Lpki
(7)
(8)
認証
暗号鍵及び
アクセス権を設定
鍵
アクセス権
TLB &
TRB:
GT (CPU)
アプリケーション
DRM
PKI ライブラリ・モジュール
←：データの流れ
うめこみ
Lxx：GT ライセンス
Cxx：証明書

図１６

記号	意味	read権	write権	execute権
PR	指定コードのみ読み出し可	指定コード*1	なし	なし
X (PRX)	指定コードからの読み込みと実行が可能	指定コード*1	なし	指定コード*1
PRW	指定コードのみ読み書き可	指定コード*1	指定コード*1	なし
PWX (PRWX)	指定コードからの読み書きと実行が可能	指定コード*1	指定コード*1	指定コード*1

図１７

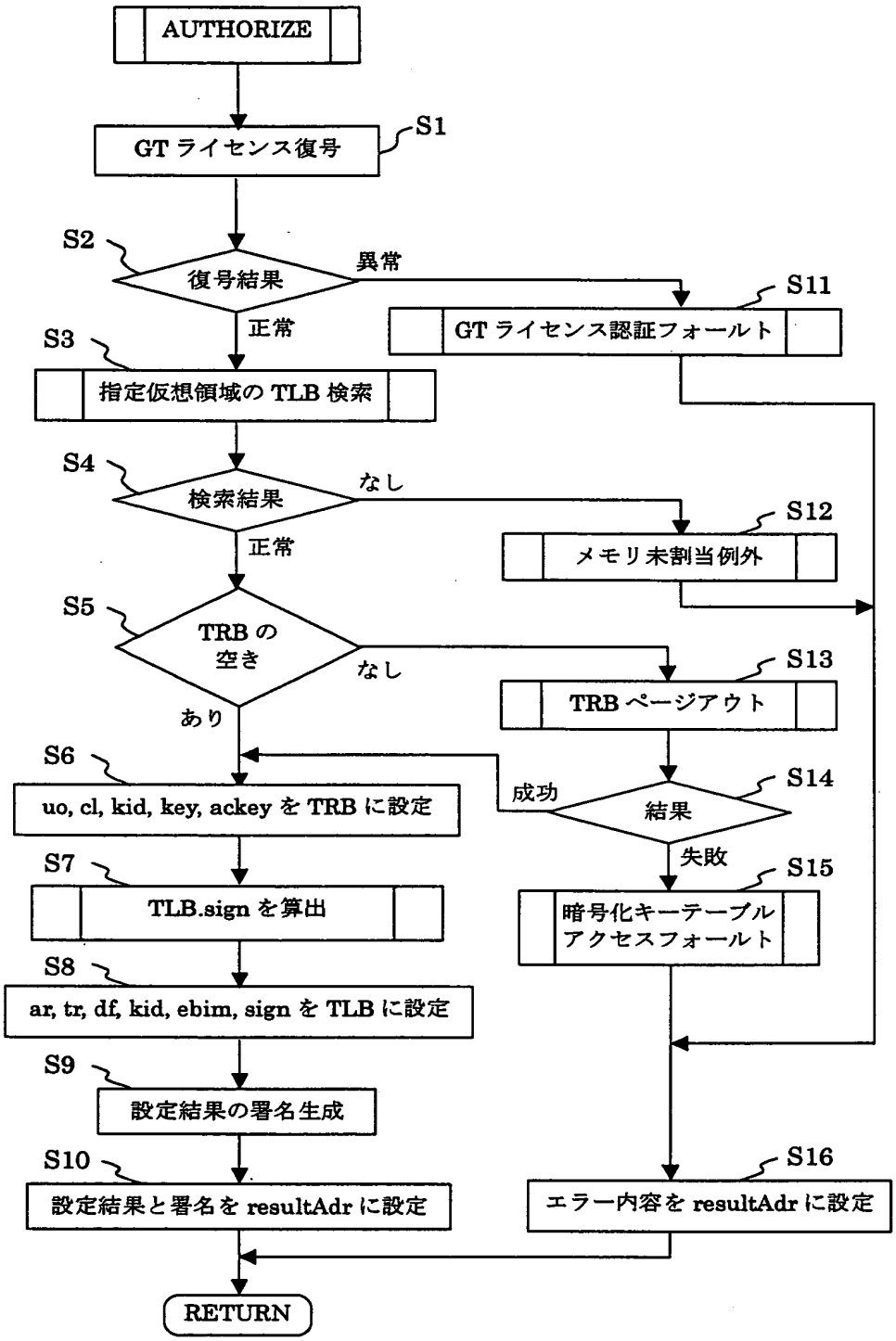
AUTHORIZE
GT ライセンス復号
S1
S2
復号結果
異常
正常
S11
GT ライセンス認証フォールト
S3
指定仮想領域の TLB 検索
S4
検索結果
なし
正常
S12
メモリ未割当例外
S5
TRB の
空き
なし
あり
S13
TRB ページアウト
S6
uo, cl, kid, key, ackey を TRB に設定
成功
S14
結果
失敗
S7
TLB.sign を算出
S15
暗号化キーテーブル
アクセスフォールト
S8
ar, tr, df, kid, ebim, sign を TLB に設定
S9
設定結果の署名生成
S10
設定結果と署名を resultAdr に設定
S16
エラー内容を resultAdr に設定
RETURN

図１８

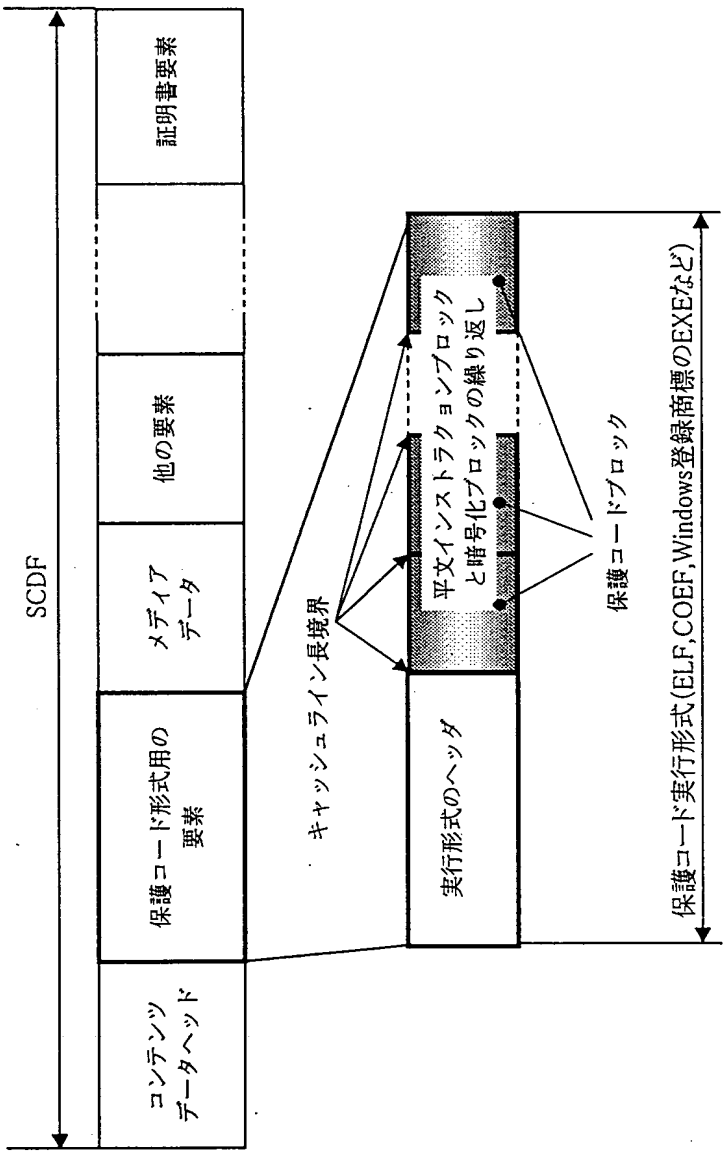
SCDF
コンテンツデータヘッド
保護コード形式用の要素
メディアデータ
他の要素
証明書要素
キャッシュライン長境界
実行形式のヘッダ
平文インストラクションブロックと暗号化ブロックの繰り返し
保護コードブロック
保護コード実行形式(ELF,COEF,Windows登録商標のEXEなど)

図１９

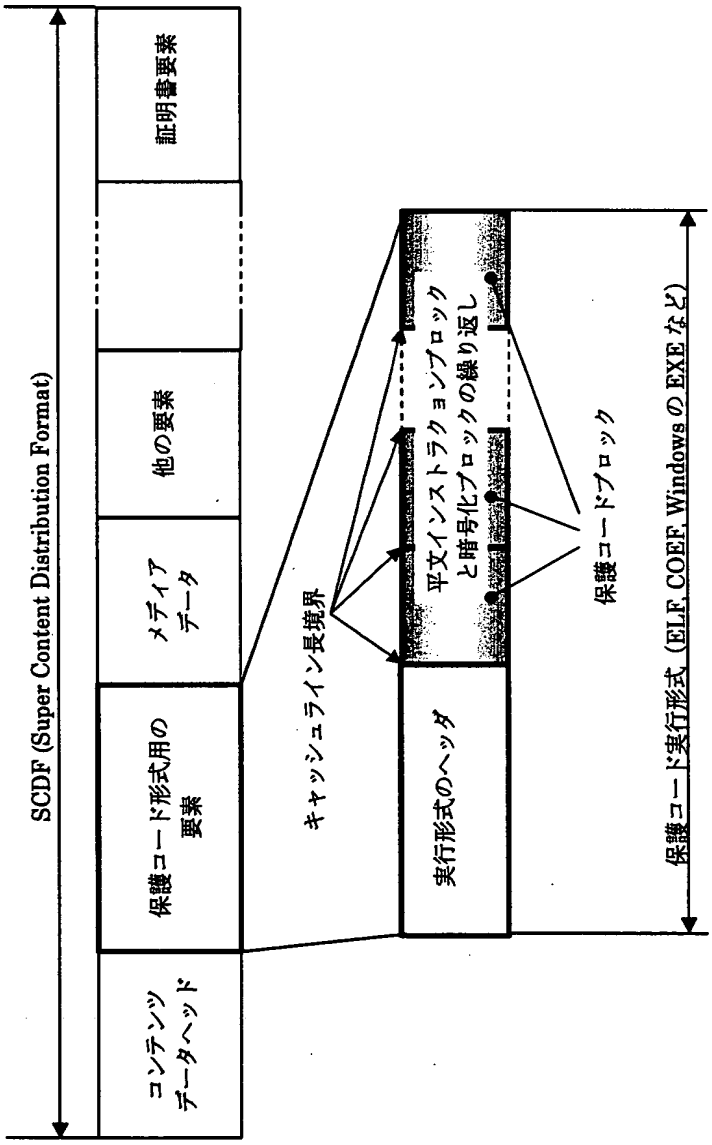
SCDF (Super Content Distribution Format)
コンテンツデータヘッド
保護コード形式用の要素
メディアデータ
他の要素
証明書要素
キャッシュライン長境界
実行形式のヘッダ
平文インストラクションブロックと暗号化ブロックの繰り返し
保護コードブロック
保護コード実行形式 (ELF, COFF, Windows の EXE など)

図１９

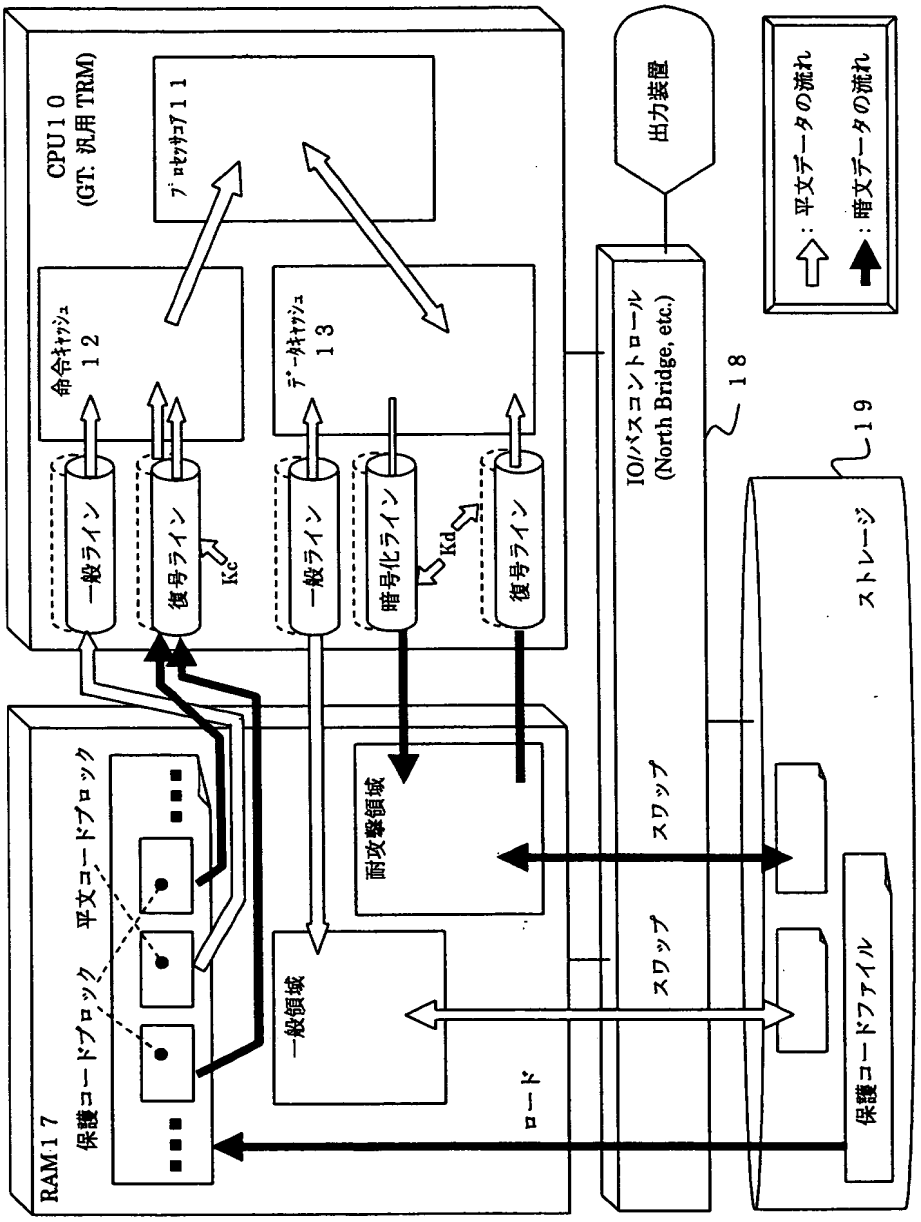
CPU10
(GT: 汎用TRM)
プロセッサコア11
命令キャッシュ
12
データキャッシュ
13
一般ライン
復号ライン
Kc
一般ライン
暗号化ライン
Kd
復号ライン
出力装置
IO/バスコントロール
(North Bridge, etc.)
18
19
ストレージ
：平文データの流れ
：暗文データの流れ
RAM17
保護コードブロック
平文コードブロック
耐攻撃領域
一般領域
ロード
スワップ
スワップ
保護コードファイル

図20

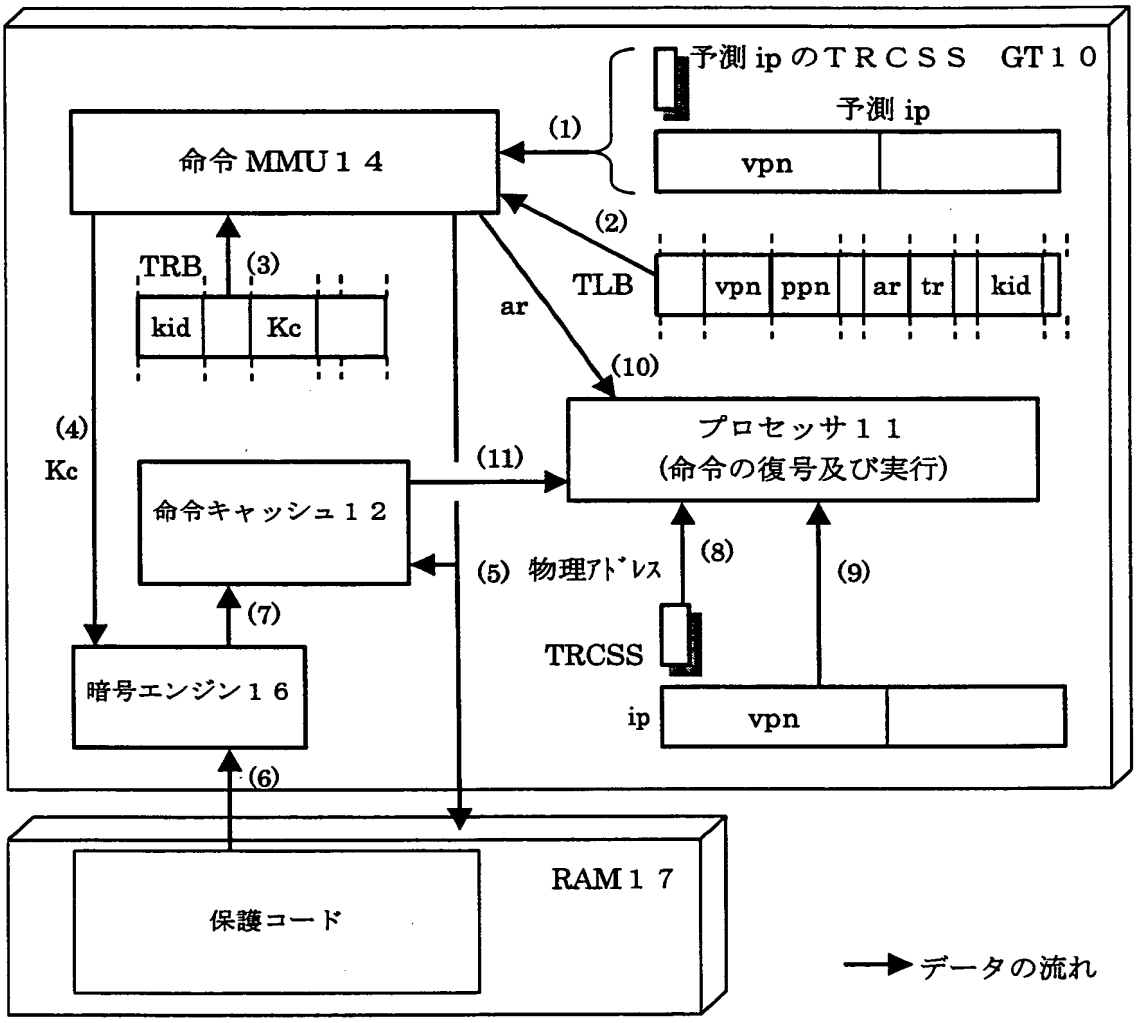
予測 ip のＴＲＣＳＳ　GT１０
予測 ip
vpn
(1)
命令 MMU１４
(2)
TRB
(3)
kid
Kc
TLB
vpn
ppn
ar
tr
kid
ar
(10)
(4)
Kc
プロセッサ１１
(命令の復号及び実行)
(11)
命令キャッシュ１２
(5) 物理ｱﾄﾞﾚｽ
(8)
(9)
(7)
TRCSS
暗号エンジン１６
ip
vpn
(6)
RAM１７
保護コード
データの流れ

図２１

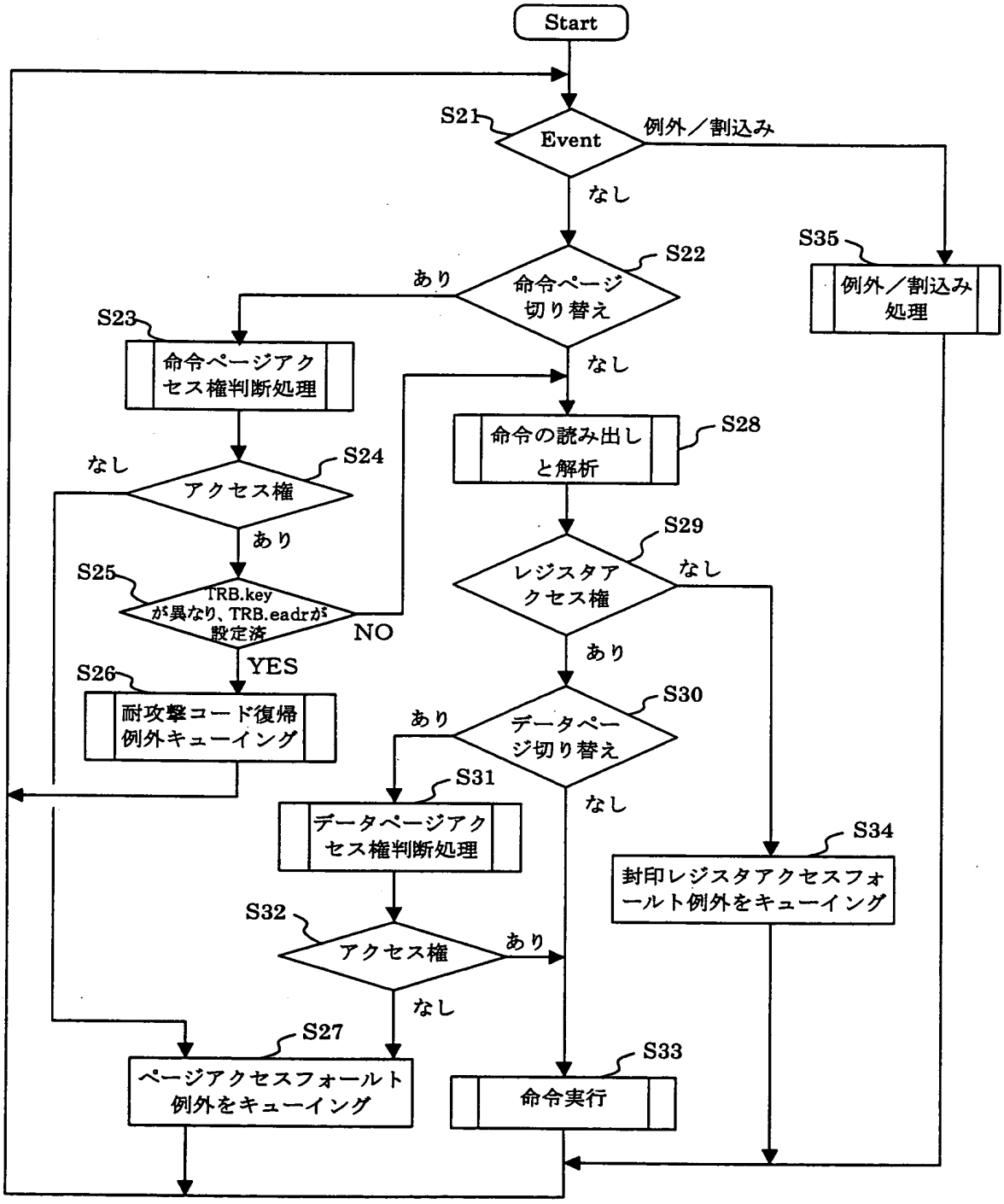
Start
S21
Event
例外／割込み
なし
S35
例外／割込み処理
S22
命令ページ切り替え
あり
S23
命令ページアクセス権判断処理
なし
S28
命令の読み出しと解析
S24
アクセス権
なし
あり
S29
レジスタアクセス権
なし
S25
TRB.keyが異なり、TRB.eadrが設定済
NO
YES
あり
S26
耐攻撃コード復帰例外キューイング
S30
データページ切り替え
あり
なし
S31
データページアクセス権判断処理
S34
封印レジスタアクセスフォールト例外をキューイング
S32
アクセス権
あり
なし
S27
ページアクセスフォールト例外をキューイング
S33
命令実行

図２２

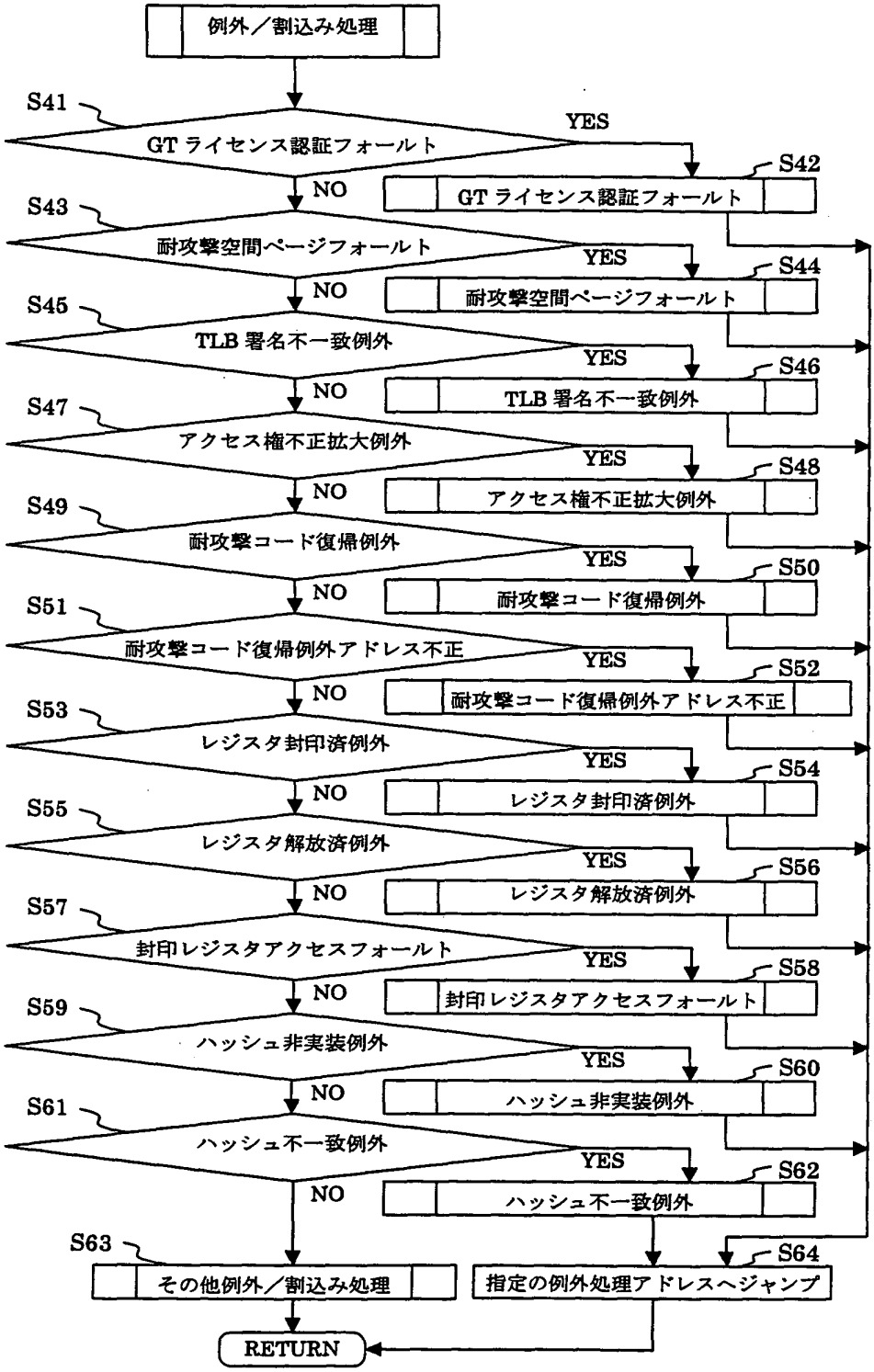
例外／割込み処理
S41
GT ライセンス認証フォールト
YES
S42
GT ライセンス認証フォールト
NO
S43
耐攻撃空間ページフォールト
YES
S44
耐攻撃空間ページフォールト
NO
S45
TLB 署名不一致例外
YES
S46
TLB 署名不一致例外
NO
S47
アクセス権不正拡大例外
YES
S48
アクセス権不正拡大例外
NO
S49
耐攻撃コード復帰例外
YES
S50
耐攻撃コード復帰例外
NO
S51
耐攻撃コード復帰例外アドレス不正
YES
S52
耐攻撃コード復帰例外アドレス不正
NO
S53
レジスタ封印済例外
YES
S54
レジスタ封印済例外
NO
S55
レジスタ解放済例外
YES
S56
レジスタ解放済例外
NO
S57
封印レジスタアクセスフォールト
YES
S58
封印レジスタアクセスフォールト
NO
S59
ハッシュ非実装例外
YES
S60
ハッシュ非実装例外
NO
S61
ハッシュ不一致例外
YES
S62
ハッシュ不一致例外
NO
S63
その他例外／割込み処理
S64
指定の例外処理アドレスへジャンプ
RETURN

図２３

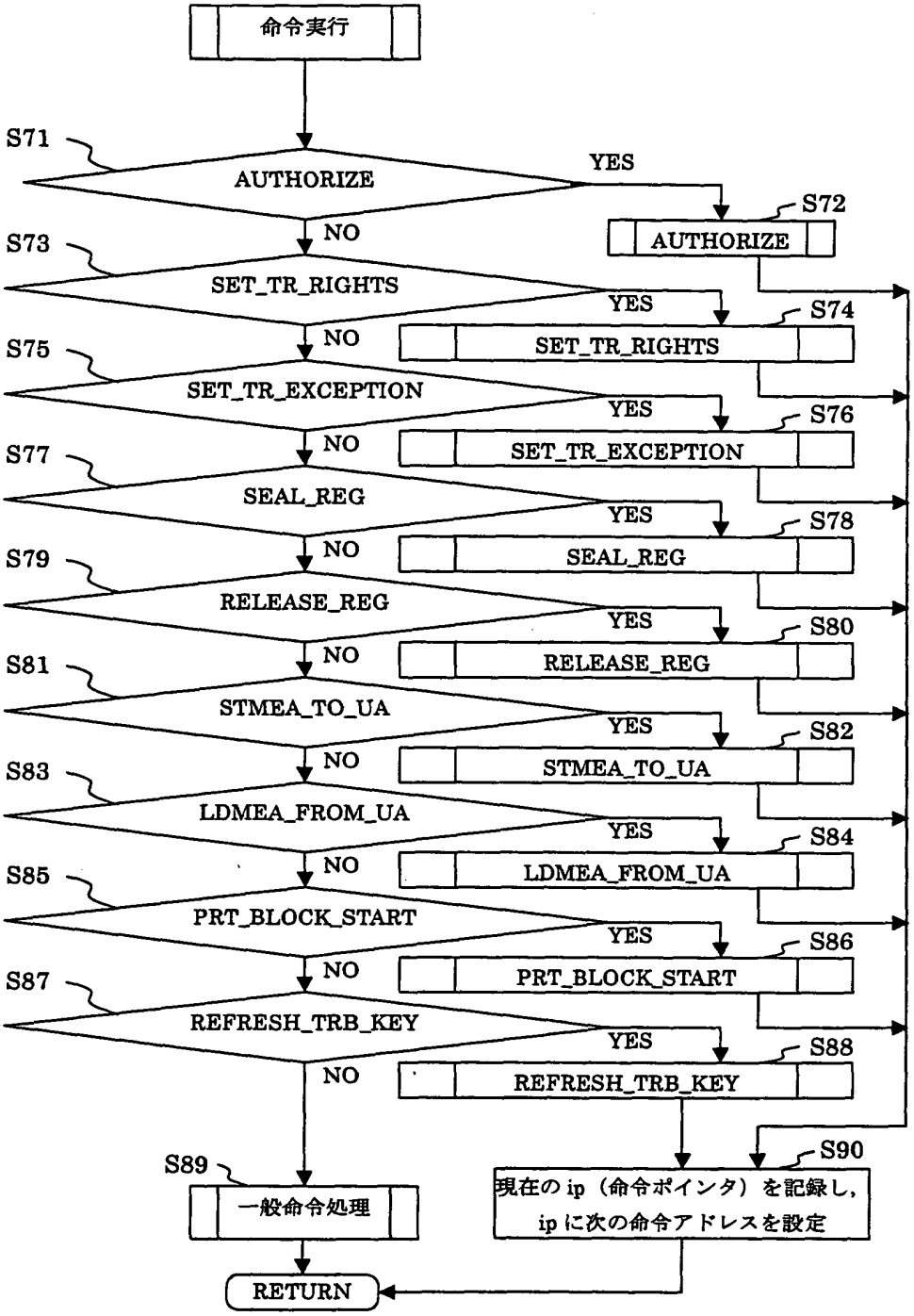
命令実行
S71
AUTHORIZE
YES
S72
AUTHORIZE
NO
S73
SET_TR_RIGHTS
YES
S74
SET_TR_RIGHTS
NO
S75
SET_TR_EXCEPTION
YES
S76
SET_TR_EXCEPTION
NO
S77
SEAL_REG
YES
S78
SEAL_REG
NO
S79
RELEASE_REG
YES
S80
RELEASE_REG
NO
S81
STMEA_TO_UA
YES
S82
STMEA_TO_UA
NO
S83
LDMEA_FROM_UA
YES
S84
LDMEA_FROM_UA
NO
S85
PRT_BLOCK_START
YES
S86
PRT_BLOCK_START
NO
S87
REFRESH_TRB_KEY
YES
S88
REFRESH_TRB_KEY
NO
S89
一般命令処理
S90
現在の ip（命令ポインタ）を記録し，
ip に次の命令アドレスを設定
RETURN

図２４

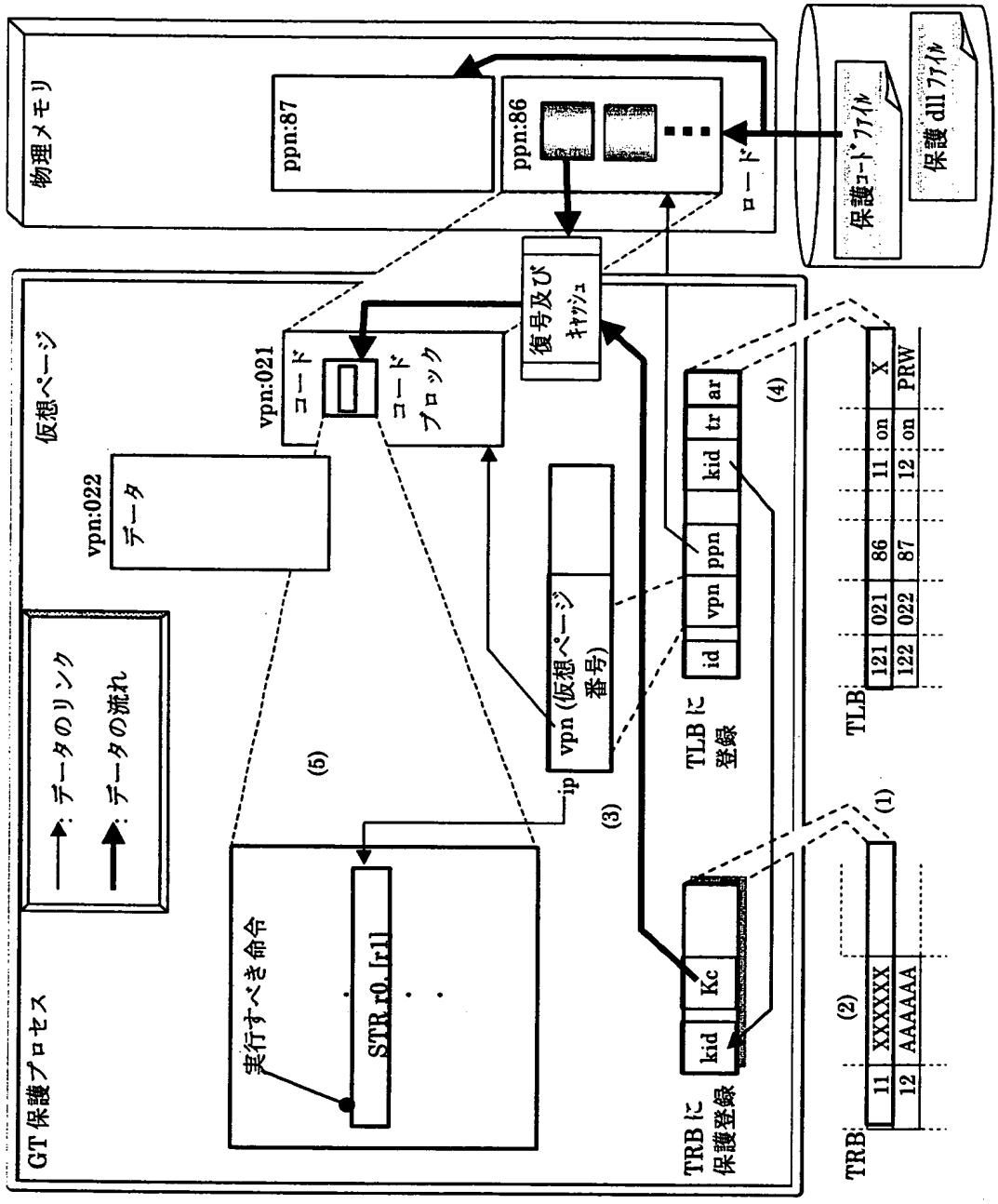
物理メモリ
ppn:87
ppn:86
ロード
保護コードファイル
保護 dll ファイル
仮想ページ
vpn:021
コード
コード
ブロック
復号及び
キャッシュ
vpn:022
データ
GT 保護プロセス
: データのリンク
: データの流れ
実行すべき命令
STR r0, [r1]
(5)
vpn (仮想ページ
番号)
ip
(3)
TLB に
登録
id
vpn
ppn
kid
tr
ar
(4)
TLB
121
122
021
022
86
87
11 : on
12 : on
X
PRW
(1)
TRB
(2)
11
12
XXXXXX
AAAAAA
TRB に
保護登録
kid
Kc

図２５

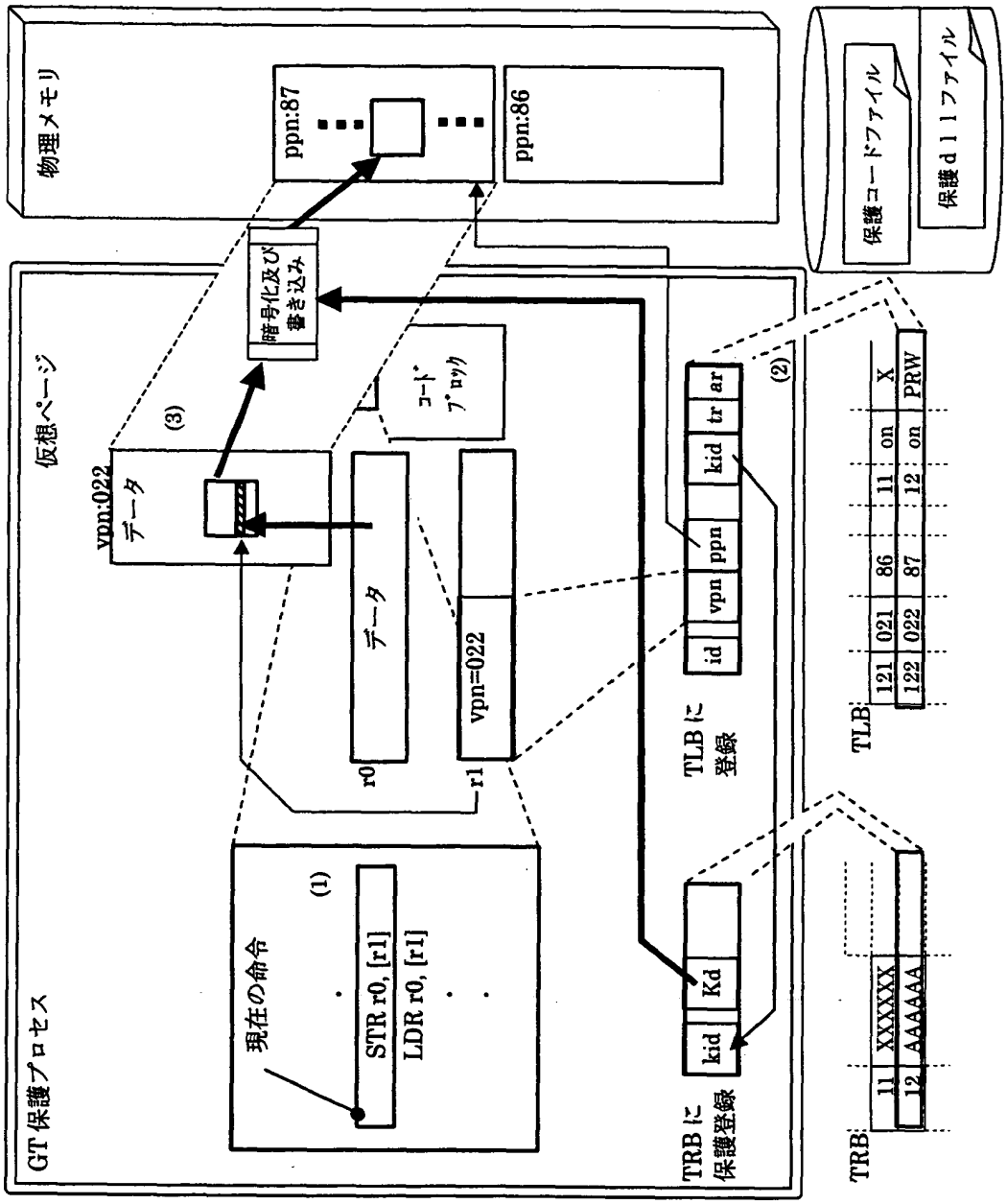
物理メモリ
ppn:87
ppn:86
保護コードファイル
保護ｄｌｌファイル
暗号化及び書き込み
GT 保護プロセス
仮想ページ
vpn:022
データ
(3)
コードブロック
データ
vpn=022
r0
r1
(1)
現在の命令
STR r0, [r1]
LDR r0, [r1]
TLB に登録
id vpn ppn kid tr ar
(2)
TLB:
121 021 86 11 on X
122 022 87 12 on PRW
TRB に保護登録
kid Kd
TRB:
11 XXXXXX
12 AAAAAA

図２６

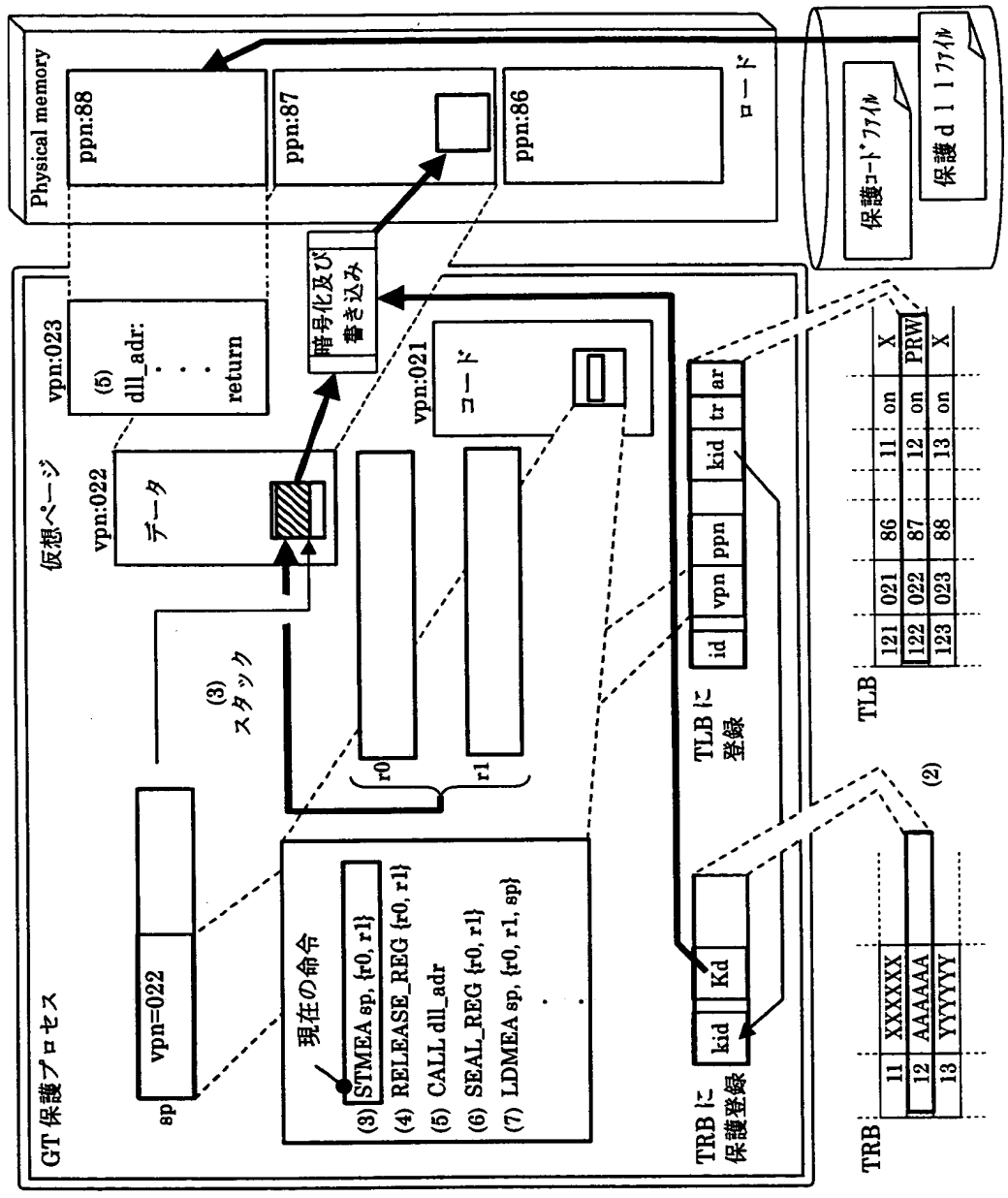
Physical memory
ppn:88
ppn:87
ppn:86
ロード
保護コードファイル
保護dllファイル
暗号化及び書き込み
仮想ページ
vpn:023
(5)
dll_adr:
return
vpn:022
データ
vpn:021
コード
(3)
スタック
r0
r1
GT 保護プロセス
vpn=022
sp
現在の命令
(3) STMEA sp, {r0, r1}
(4) RELEASE_REG {r0, r1}
(5) CALL dll_adr
(6) SEAL_REG {r0, r1}
(7) LDMEA sp, {r0, r1, sp}
TLBに登録
id
vpn
ppn
kid
tr
ar
TLB
121 021 86 11 on X
122 022 87 12 on PRW
123 023 88 13 on X
TRBに保護登録
kid
Kd
TRB
11 XXXXXX
12 AAAAAA
13 YYYYYY
(2)

図27

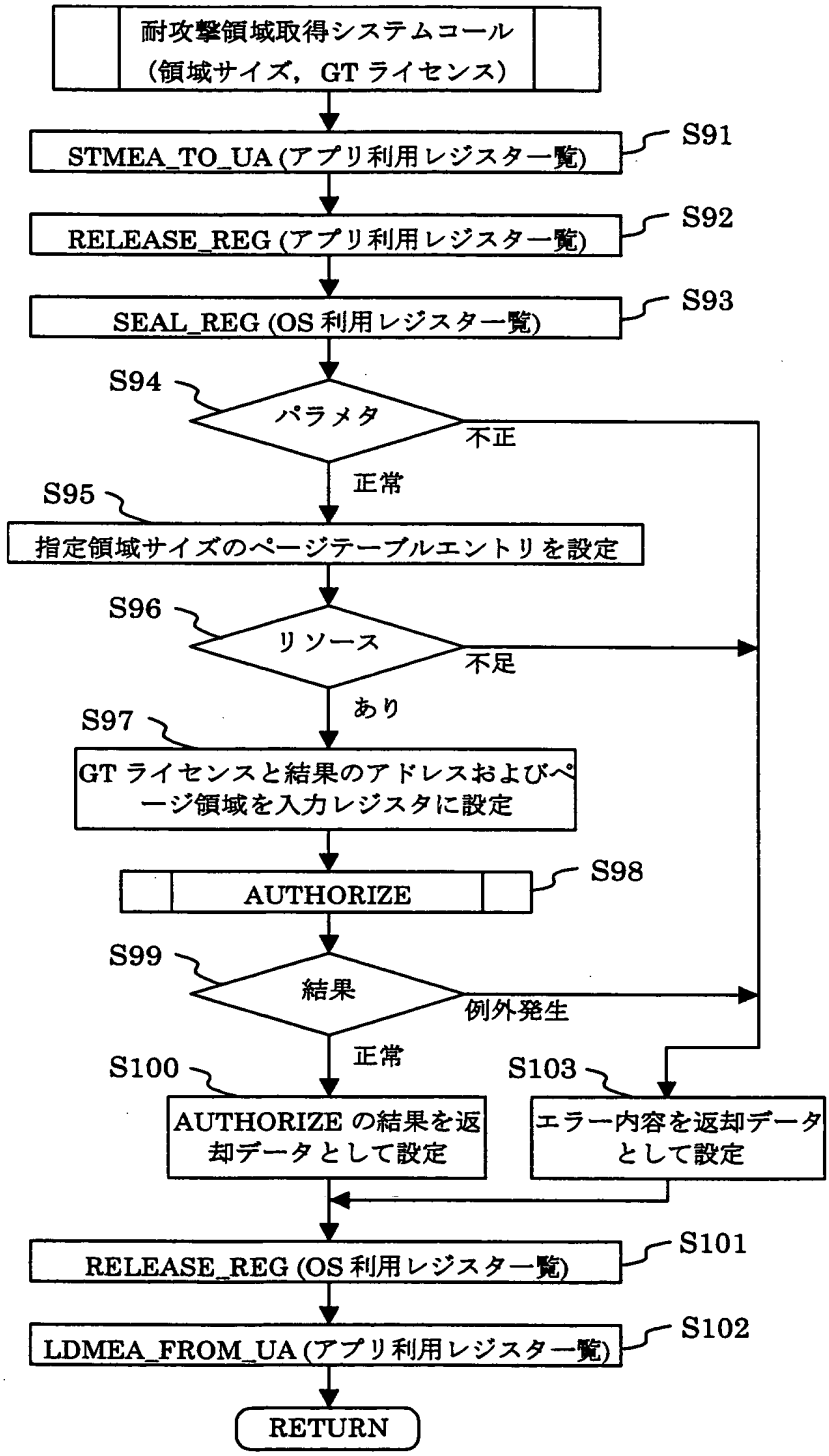
耐攻撃領域取得システムコール
（領域サイズ，GT ライセンス）
STMEA_TO_UA (アプリ利用レジスター覧)
S91
RELEASE_REG (アプリ利用レジスター覧)
S92
SEAL_REG (OS 利用レジスター覧)
S93
S94
パラメタ
不正
正常
S95
指定領域サイズのページテーブルエントリを設定
S96
リソース
不足
あり
S97
GT ライセンスと結果のアドレスおよびページ領域を入力レジスタに設定
AUTHORIZE
S98
S99
結果
例外発生
正常
S100
AUTHORIZE の結果を返却データとして設定
S103
エラー内容を返却データとして設定
RELEASE_REG (OS 利用レジスター覧)
S101
LDMEA_FROM_UA (アプリ利用レジスター覧)
S102
RETURN

図２８

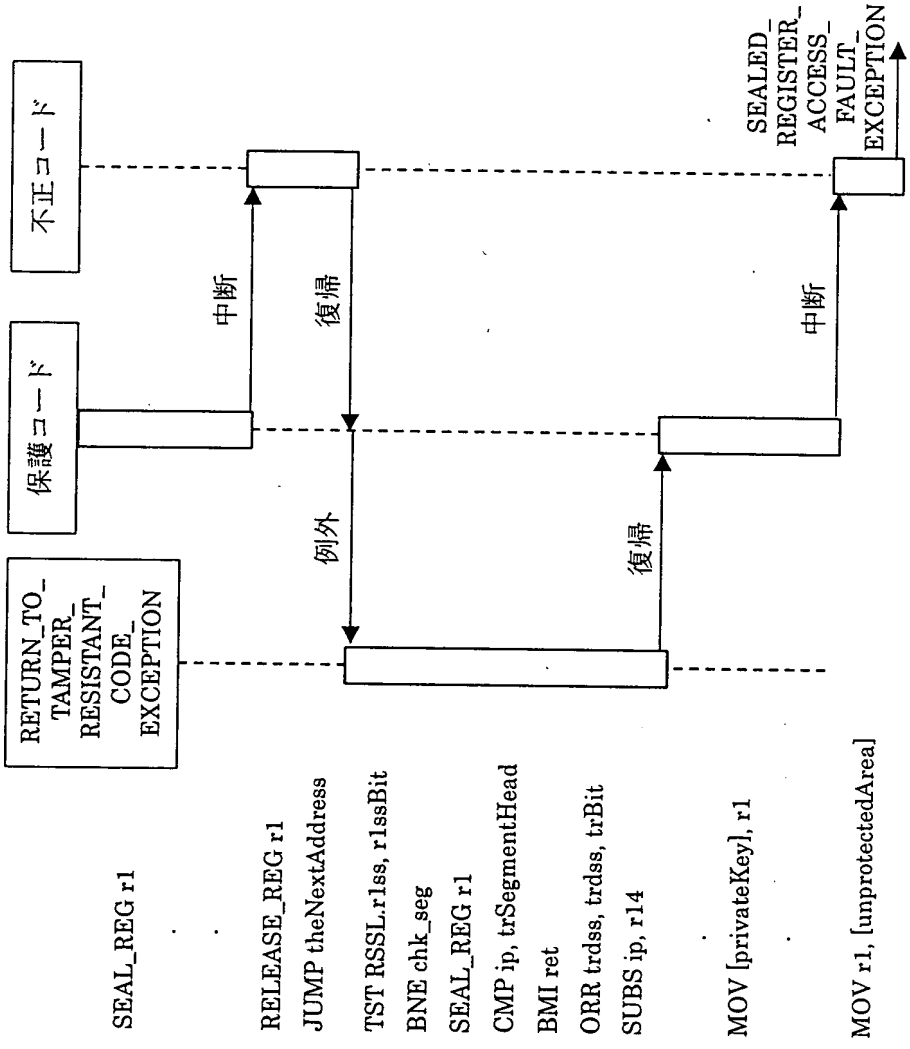
不正コード
保護コード
RETURN_TO_TAMPER_RESISTANT_CODE_EXCEPTION
SEALED_REGISTER_ACCESS_FAULT_EXCEPTION
中断
復帰
例外
復帰
中断
SEAL_REG r1
RELEASE_REG r1
JUMP theNextAddress
TST RSSL.r1ss, r1ssBit
BNE chk_seg
SEAL_REG r1
chk_seg CMP ip, trSegmentHead
BMI ret
ORR trdss, trdss, trBit
ret SUBS ip, r14
MOV [privateKey], r1
MOV r1, [unprotectedArea]

図２９

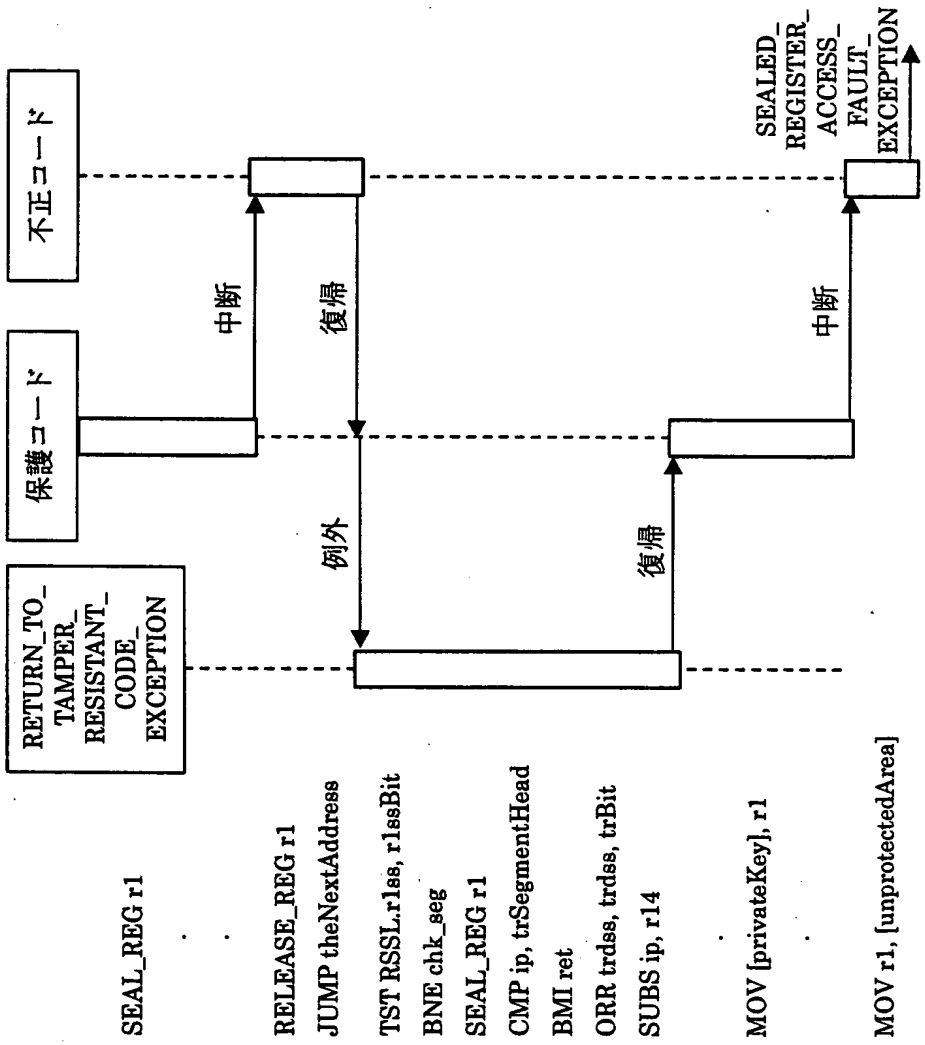
RETURN_TO_TAMPER_RESISTANT_CODE_EXCEPTION
保護コード
不正コード
中断
復帰
例外
復帰
中断
SEALED_REGISTER_ACCESS_FAULT_EXCEPTION
SEAL_REG r1
RELEASE_REG r1
JUMP theNextAddress
TST RSSL.r1ss, r1ssBit
BNE chk_seg
SEAL_REG r1
chk_seg CMP ip, trSegmentHead
BMI ret
ORR trdss, trdss, trBit
ret SUBS ip, r14
MOV [privateKey], r1
MOV r1, [unprotectedArea]

図29

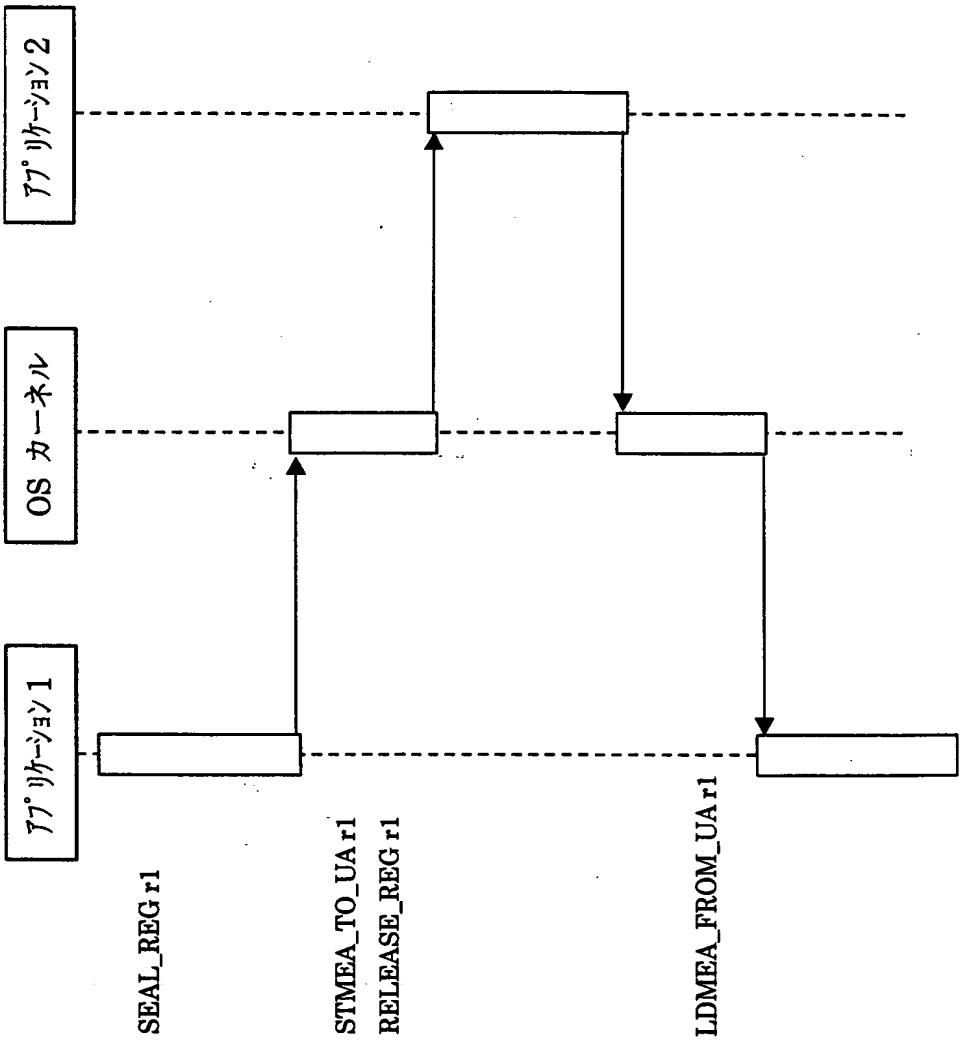
アプリケーション2
OS カーネル
アプリケーション1
SEAL_REG r1
STMEA_TO_UA r1
RELEASE_REG r1
LDMEA_FROM_UA r1

図30

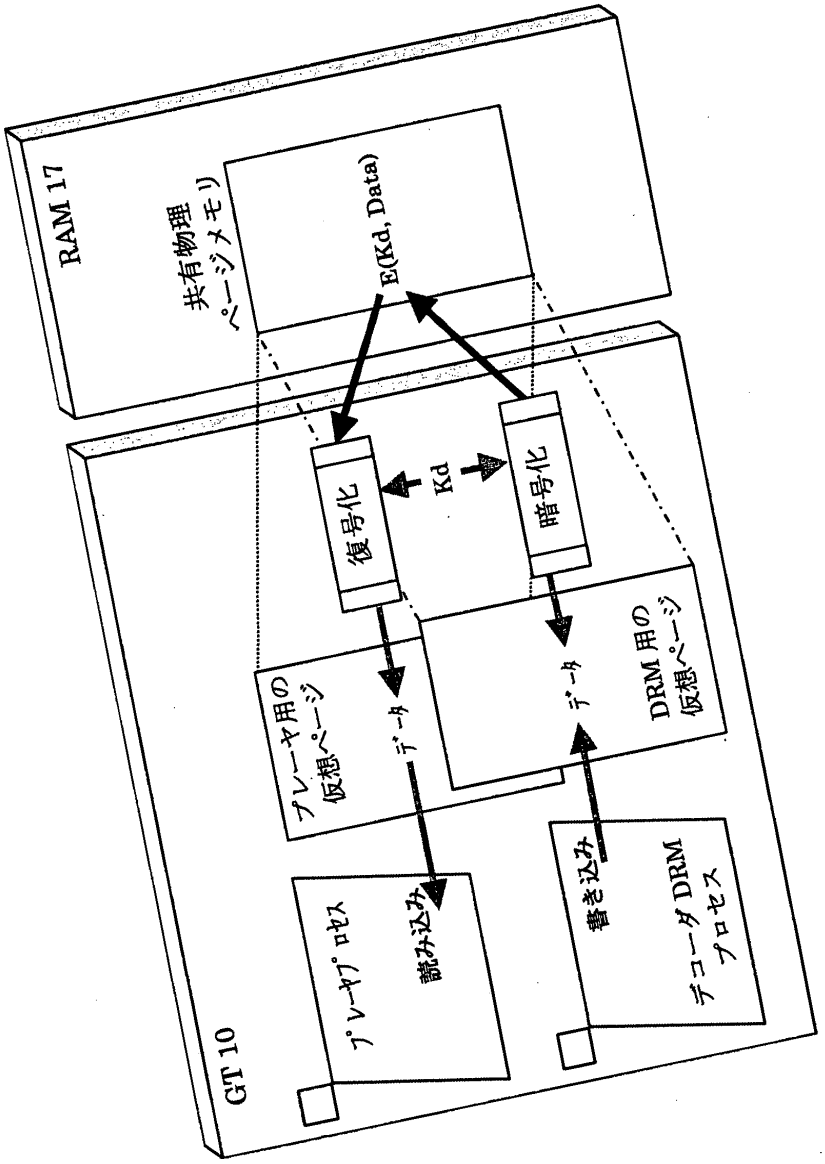
RAM 17
共有物理
ページメモリ
E(Kd, Data)
復号化
Kd
暗号化
プレーヤ用の
仮想ページ
データ
データ
DRM 用の
仮想ページ
GT 10
プレーヤプロセス
読み込み
書き込み
デコーダ DRM
プロセス

図31

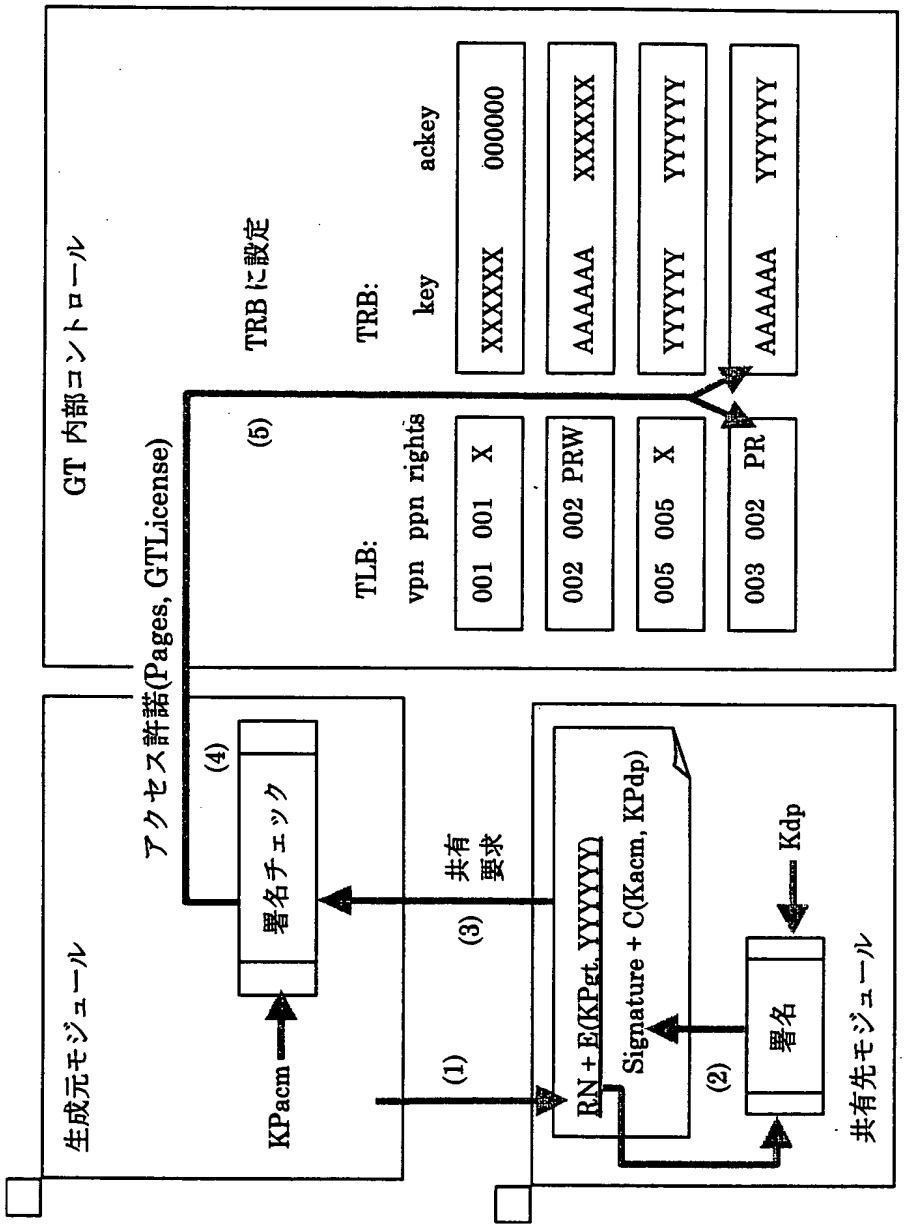
生成元モジュール
KPacm
(4)
署名チェック
アクセス許諾(Pages, GTLicense)
(1)
(3)
共有
要求
GT 内部コントロール
(5)
TRB に設定
TLB:
vpn ppn rights
001 001 X
002 002 PRW
005 005 X
003 002 PR
TRB:
key ackey
XXXXXX 000000
AAAAAA XXXXXX
YYYYYY YYYYYY
AAAAAA YYYYYY
RN + E(KPgt, YYYYYY)
Signature + C(Kacm, KPdp)
(2)
署名
Kdp
共有先モジュール

図３２

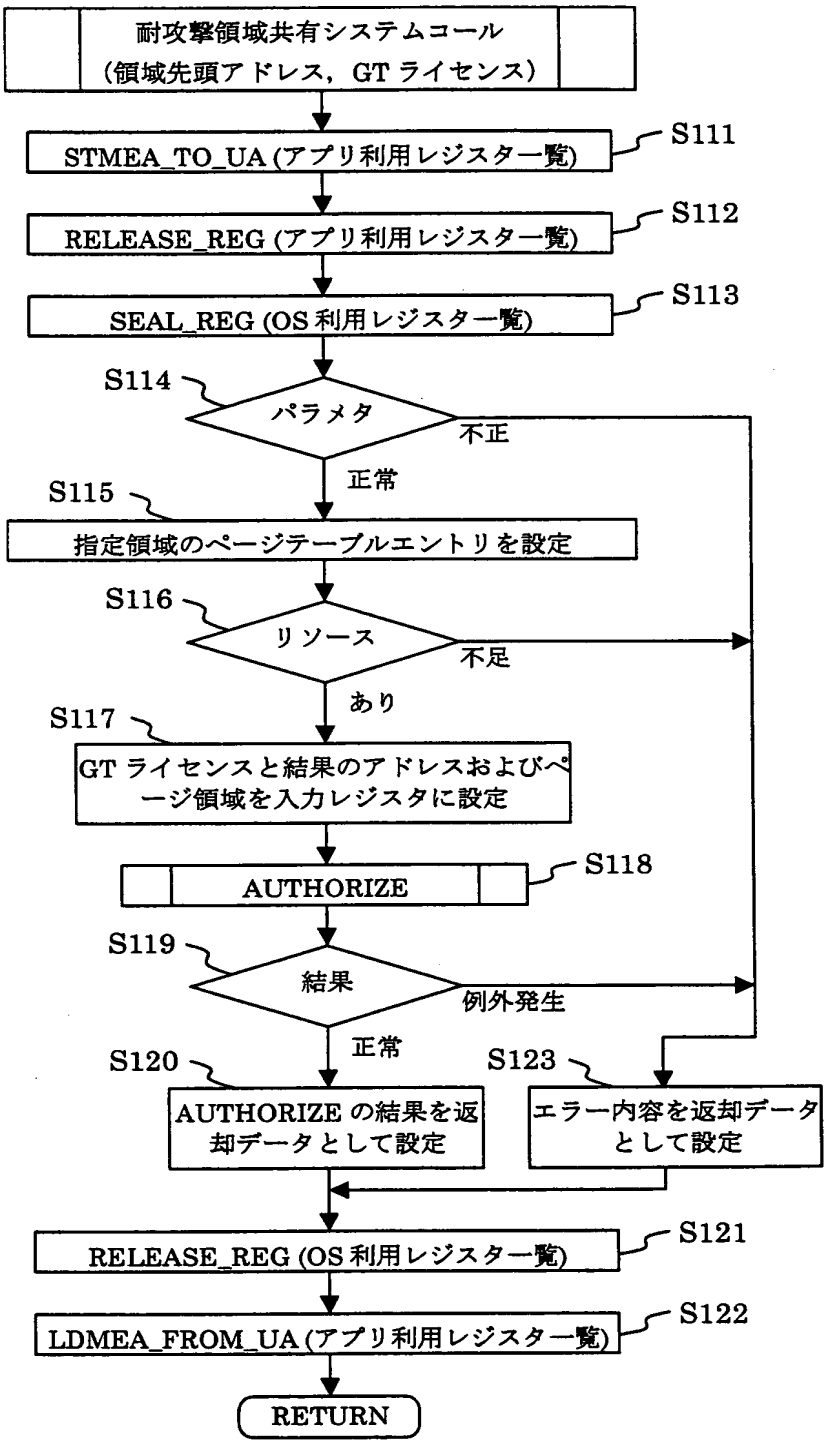
耐攻撃領域共有システムコール
(領域先頭アドレス，GT ライセンス)
STMEA_TO_UA (アプリ利用レジスター覧)
S111
RELEASE_REG (アプリ利用レジスター覧)
S112
SEAL_REG (OS 利用レジスター覧)
S113
S114
パラメタ
不正
正常
S115
指定領域のページテーブルエントリを設定
S116
リソース
不足
あり
S117
GT ライセンスと結果のアドレスおよびページ領域を入力レジスタに設定
AUTHORIZE
S118
S119
結果
例外発生
正常
S120
AUTHORIZE の結果を返却データとして設定
S123
エラー内容を返却データとして設定
RELEASE_REG (OS 利用レジスター覧)
S121
LDMEA_FROM_UA (アプリ利用レジスター覧)
S122
RETURN

図３３

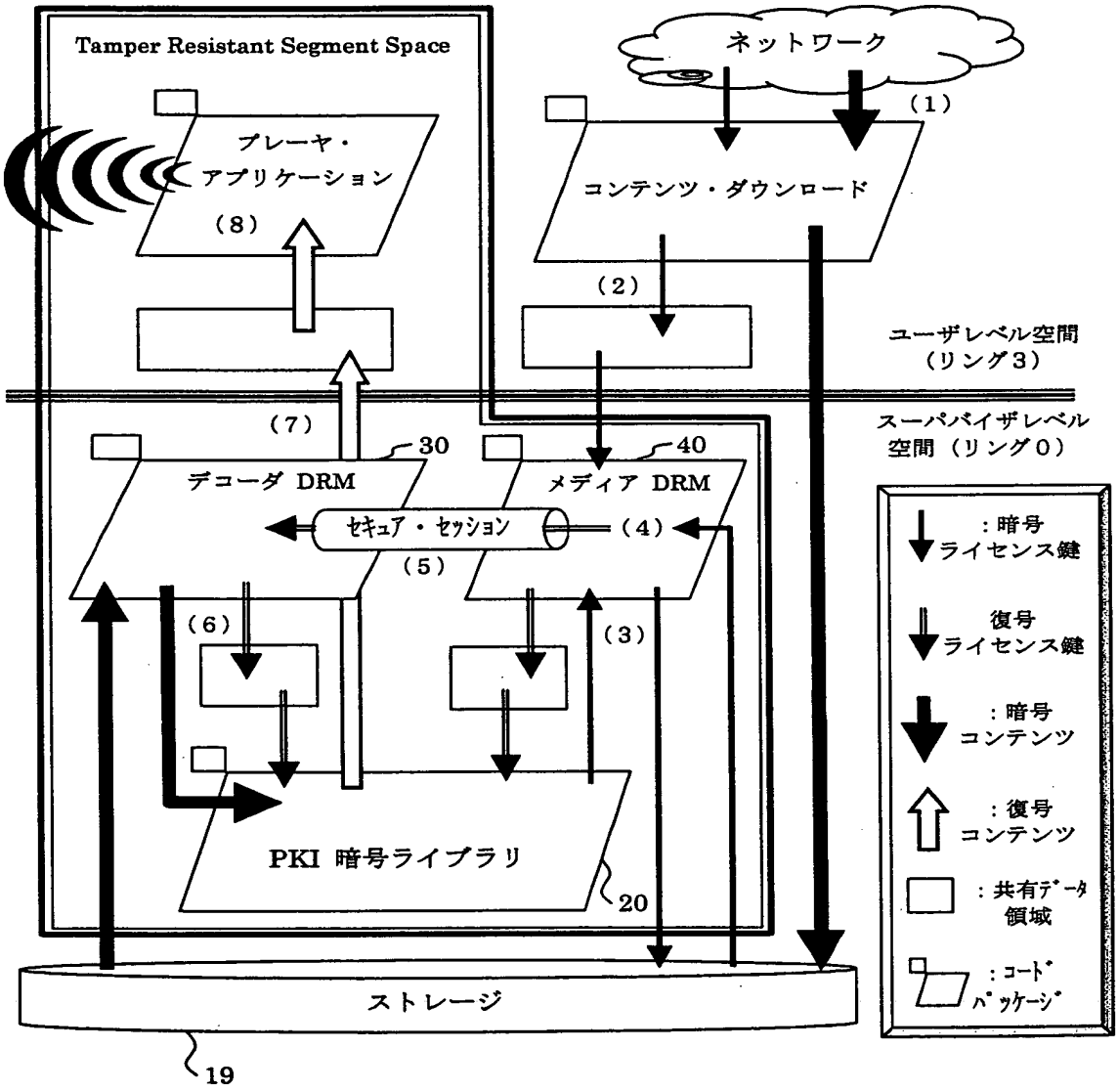
Tamper Resistant Segment Space
ネットワーク
(1)
プレーヤ・
アプリケーション
(8)
コンテンツ・ダウンロード
(2)
ユーザレベル空間
(リング3)
スーパバイザレベル
空間（リング0）
(7)
30
40
デコーダ DRM
メディア DRM
セキュア・セッション
(4)
(5)
(6)
(3)
PKI 暗号ライブラリ
20
ストレージ
19
：暗号
ライセンス鍵
復号
ライセンス鍵
：暗号
コンテンツ
：復号
コンテンツ
：共有ﾃﾞｰﾀ
領域
：ｺｰﾄﾞ
ﾊﾟｯｹｰｼﾞ

図３４

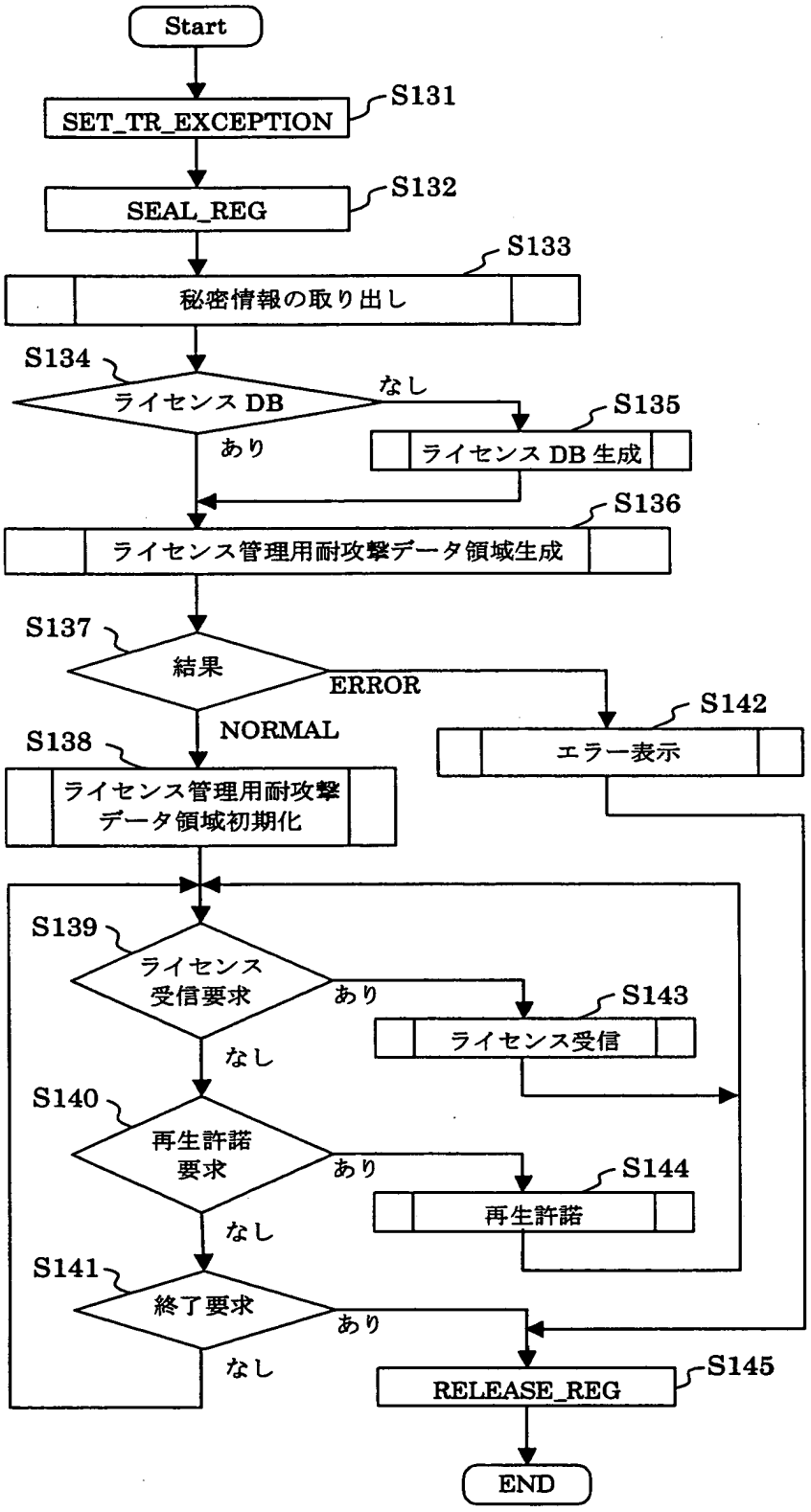
Start
SET_TR_EXCEPTION
S131
SEAL_REG
S132
S133
秘密情報の取り出し
S134
ライセンス DB
なし
あり
S135
ライセンス DB 生成
S136
ライセンス管理用耐攻撃データ領域生成
S137
結果
ERROR
NORMAL
S142
エラー表示
S138
ライセンス管理用耐攻撃
データ領域初期化
S139
ライセンス
受信要求
あり
S143
ライセンス受信
なし
S140
再生許諾
要求
あり
S144
再生許諾
なし
S141
終了要求
あり
なし
S145
RELEASE_REG
END

図３５

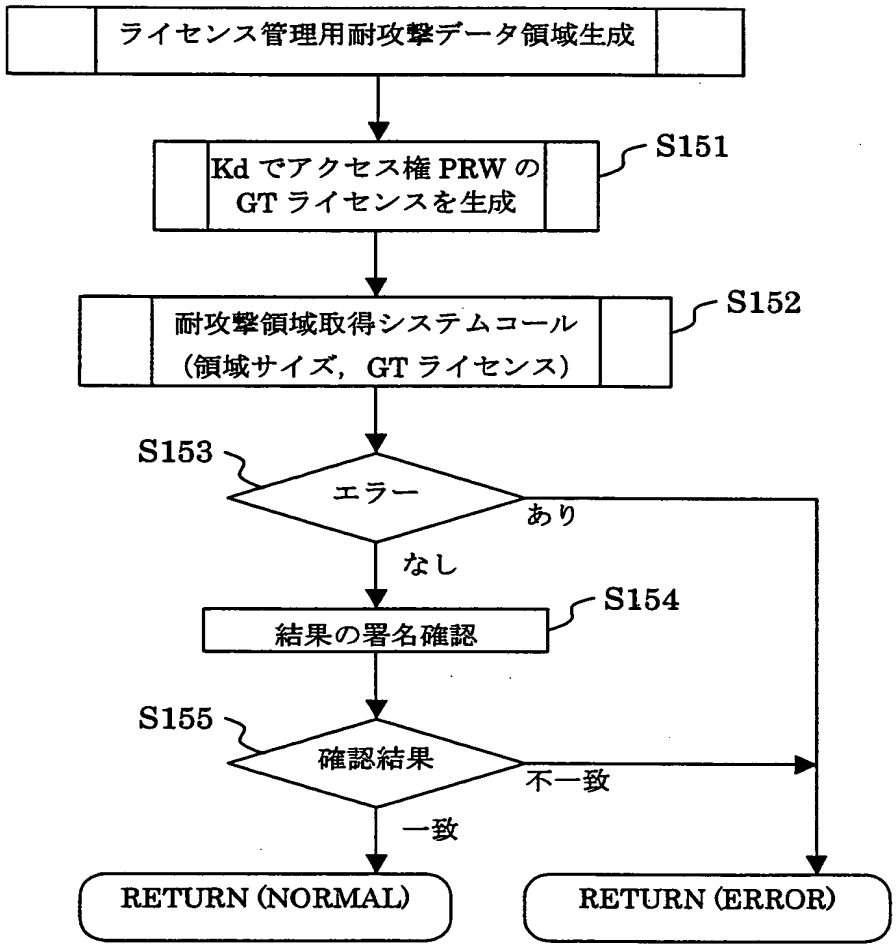
ライセンス管理用耐攻撃データ領域生成
Kd でアクセス権 PRW の GT ライセンスを生成
S151
耐攻撃領域取得システムコール（領域サイズ，GT ライセンス）
S152
S153
エラー
あり
なし
結果の署名確認
S154
S155
確認結果
不一致
一致
RETURN (NORMAL)
RETURN (ERROR)

図３６

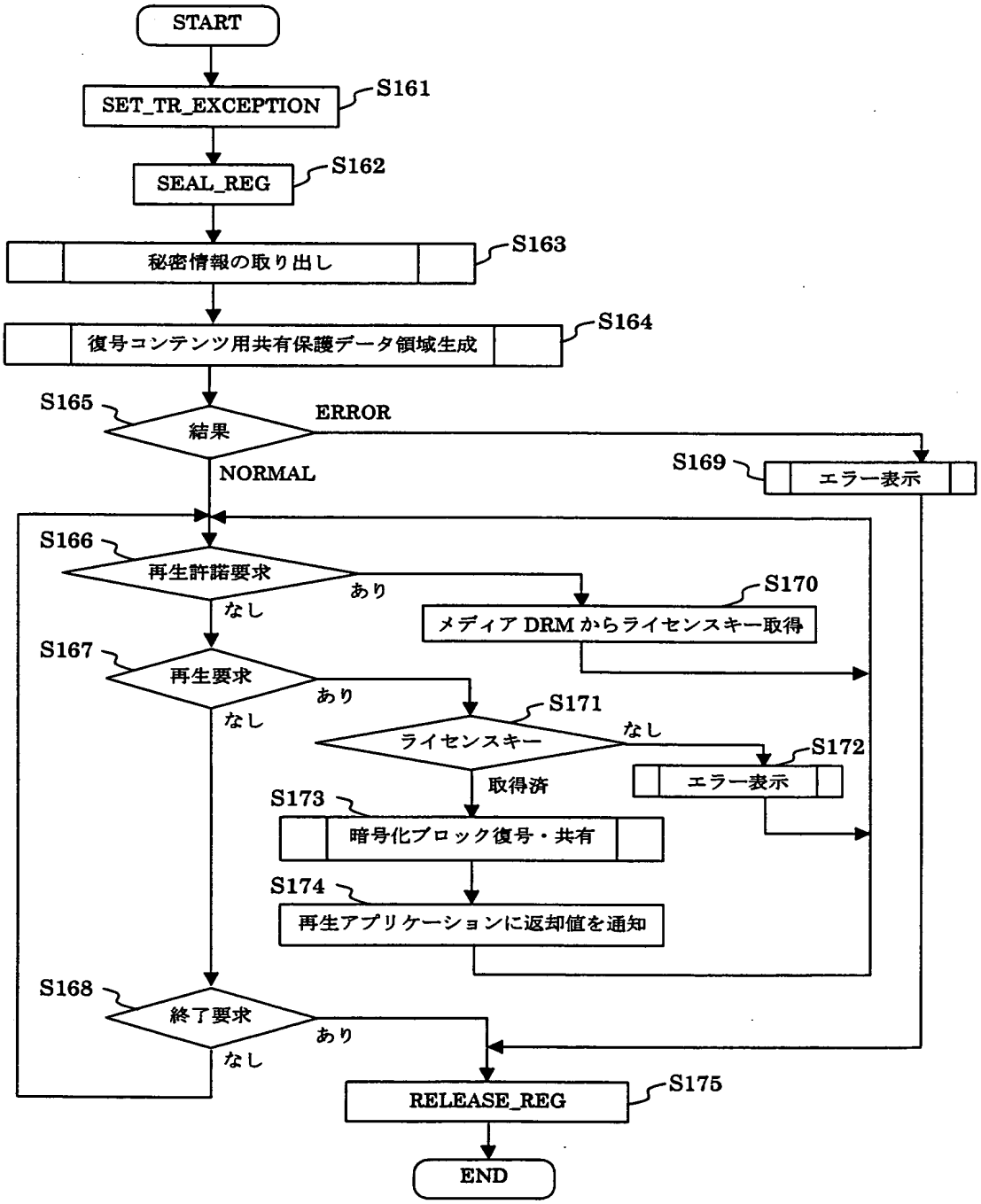
START
SET_TR_EXCEPTION
S161
SEAL_REG
S162
秘密情報の取り出し
S163
復号コンテンツ用共有保護データ領域生成
S164
S165
結果
ERROR
NORMAL
S169
エラー表示
S166
再生許諾要求
あり
なし
S170
メディア DRM からライセンスキー取得
S167
再生要求
あり
なし
S171
ライセンスキー
なし
取得済
S172
エラー表示
S173
暗号化ブロック復号・共有
S174
再生アプリケーションに返却値を通知
S168
終了要求
あり
なし
RELEASE_REG
S175
END

図３７

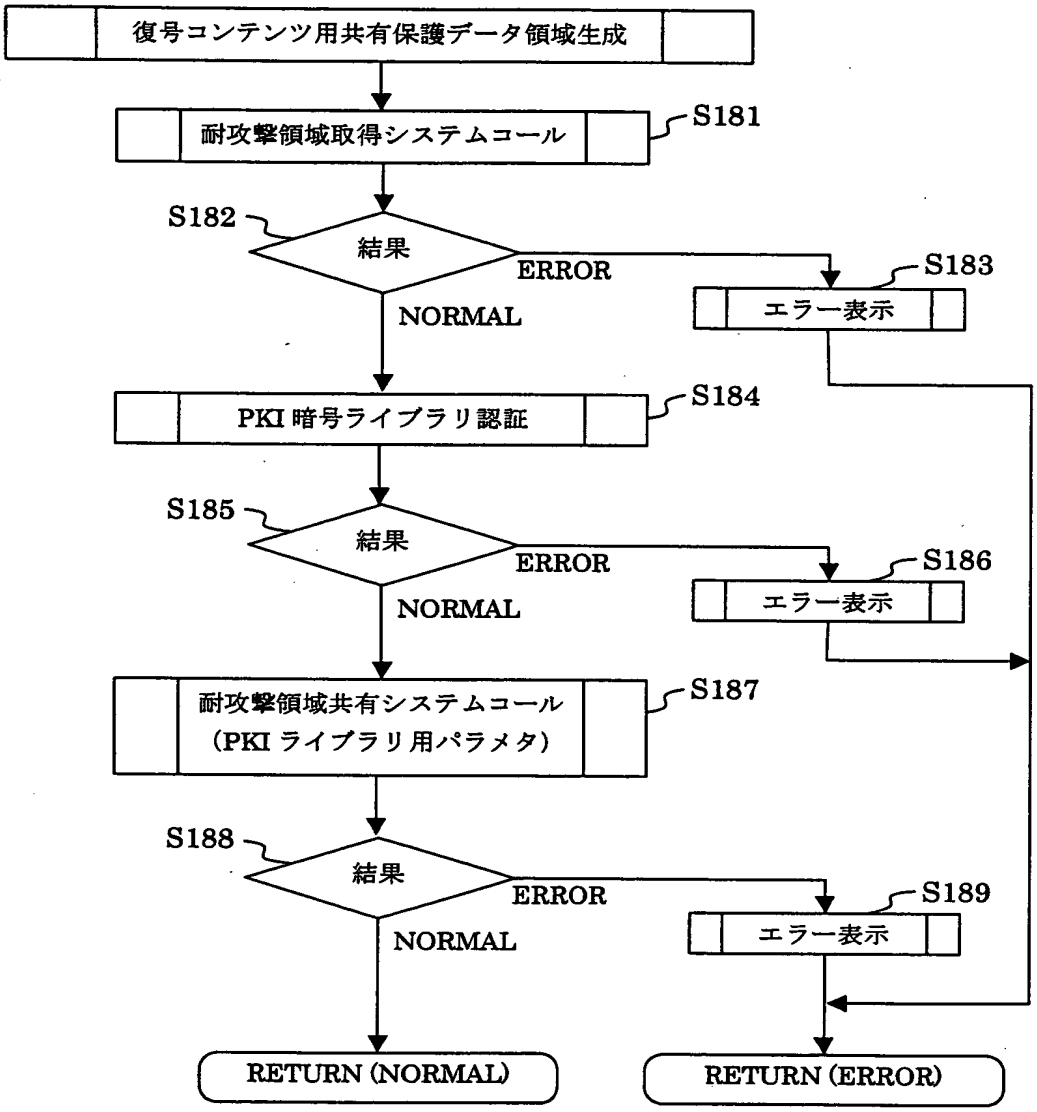
復号コンテンツ用共有保護データ領域生成
耐攻撃領域取得システムコール
S181
S182
結果
ERROR
S183
エラー表示
NORMAL
PKI 暗号ライブラリ認証
S184
S185
結果
ERROR
S186
エラー表示
NORMAL
耐攻撃領域共有システムコール
（PKI ライブラリ用パラメタ）
S187
S188
結果
ERROR
S189
エラー表示
NORMAL
RETURN (NORMAL)
RETURN (ERROR)

図３８

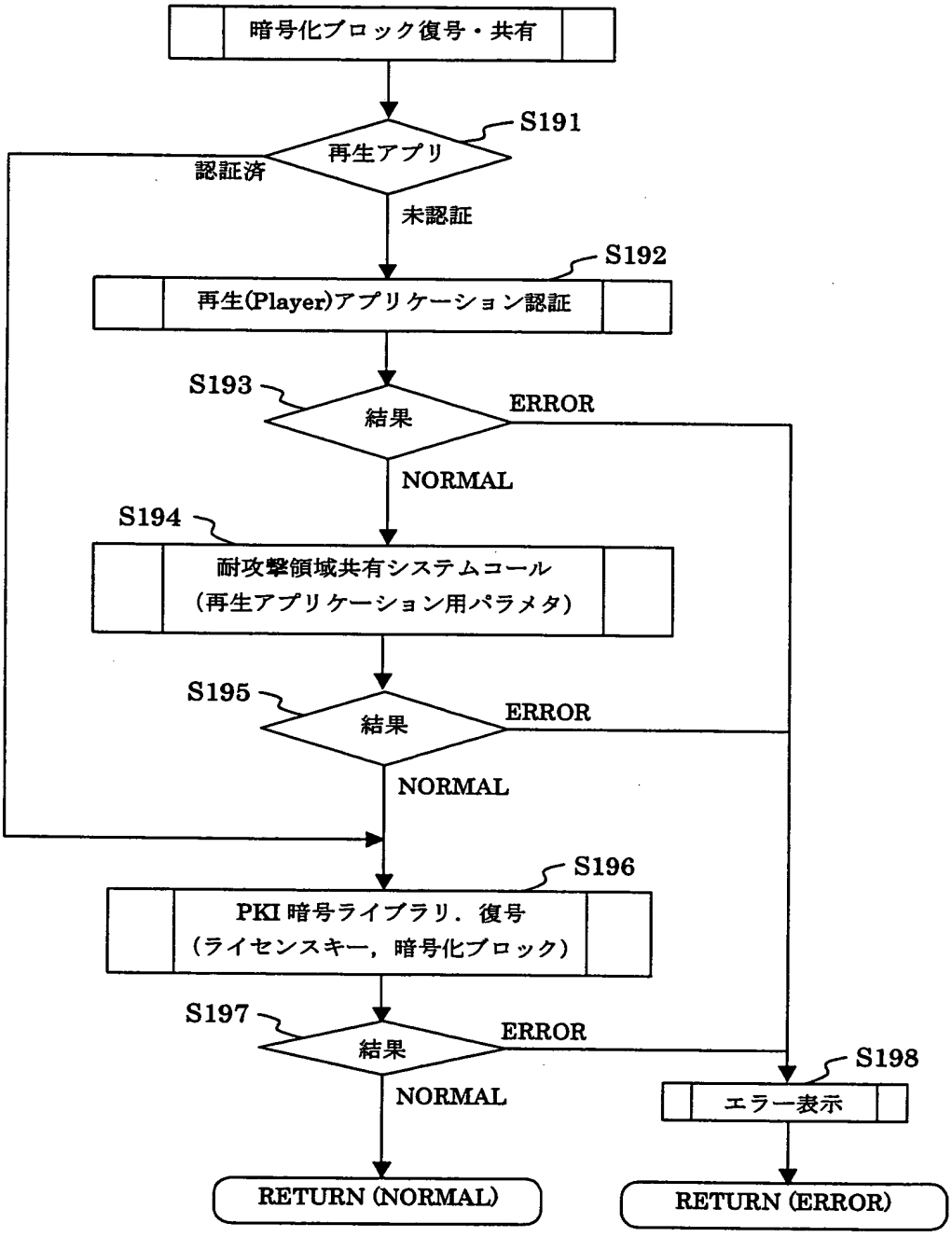
暗号化ブロック復号・共有
S191
再生アプリ
認証済
未認証
S192
再生(Player)アプリケーション認証
S193
結果
ERROR
NORMAL
S194
耐攻撃領域共有システムコール
（再生アプリケーション用パラメタ）
S195
結果
ERROR
NORMAL
S196
PKI 暗号ライブラリ．復号
（ライセンスキー，暗号化ブロック）
S197
結果
ERROR
NORMAL
S198
エラー表示
RETURN (NORMAL)
RETURN (ERROR)

図３９

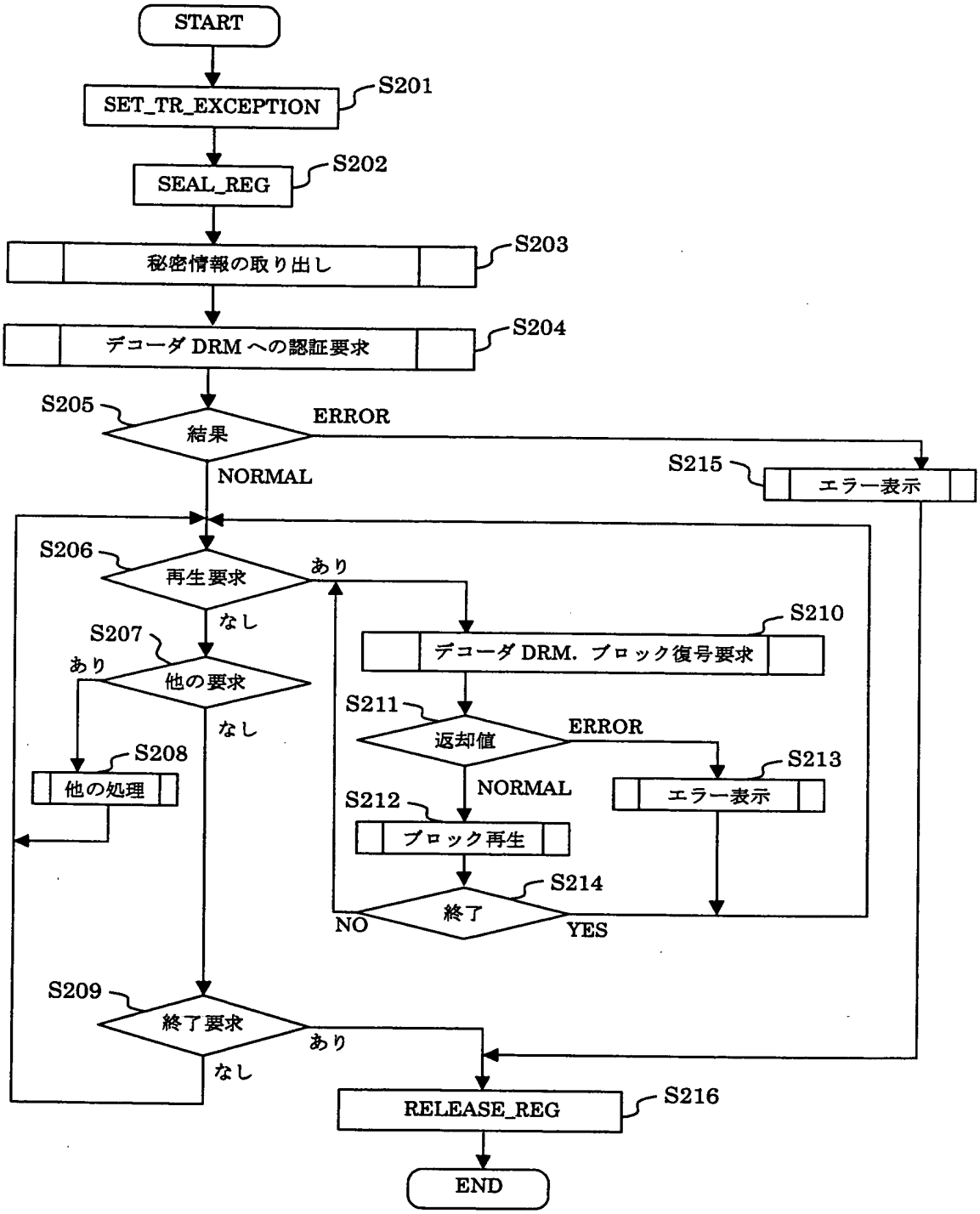
START
SET_TR_EXCEPTION
S201
SEAL_REG
S202
秘密情報の取り出し
S203
デコーダ DRM への認証要求
S204
S205
結果
ERROR
NORMAL
S215
エラー表示
S206
再生要求
あり
なし
S207
あり
他の要求
なし
S208
他の処理
S210
デコーダ DRM. ブロック復号要求
S211
返却値
ERROR
S213
エラー表示
NORMAL
S212
ブロック再生
S214
終了
NO
YES
S209
終了要求
あり
なし
RELEASE_REG
S216
END

図４０

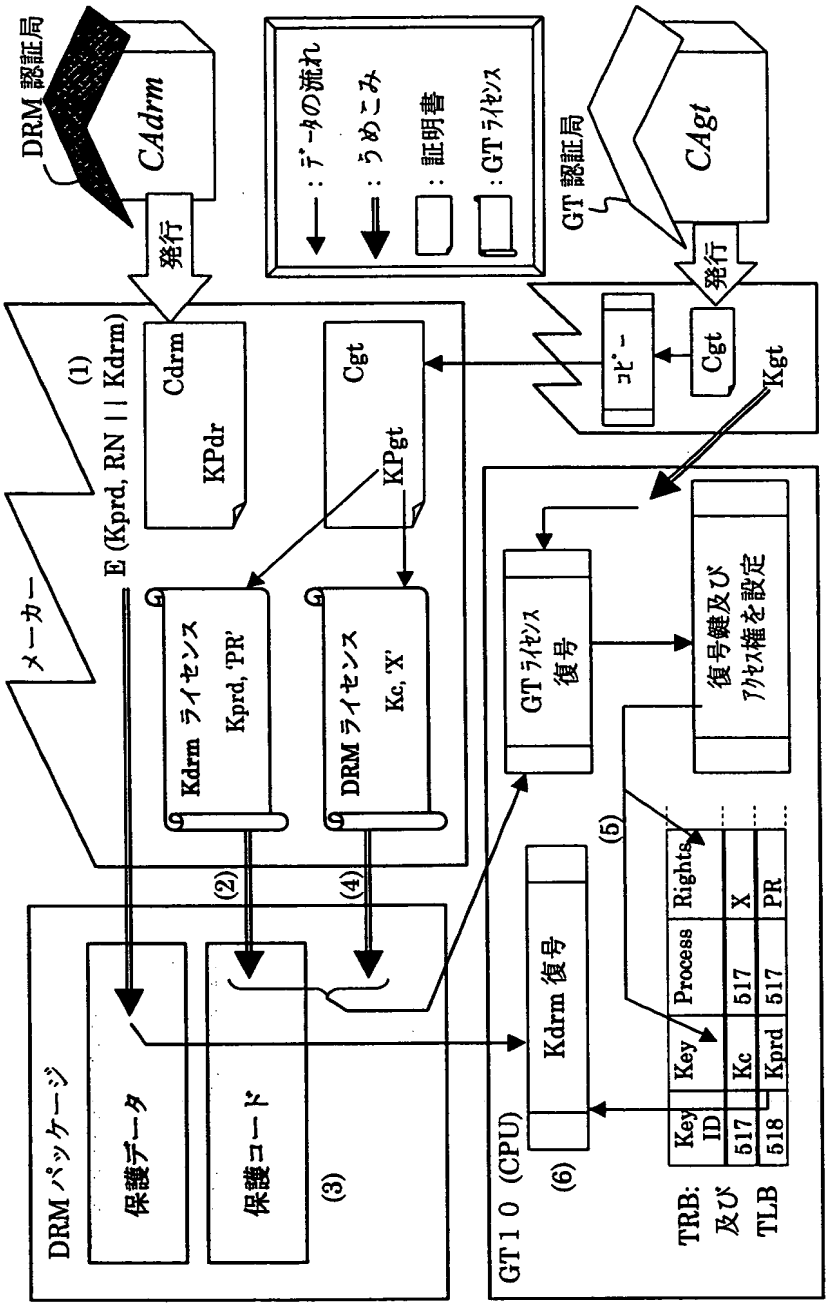
DRM 認証局
CAdrm
発行
: データの流れ
: うめこみ
: 証明書
: GT ライセンス
GT 認証局
CAgt
発行
(1)
E (Kprd, RN | | Kdrm)
Cdrm
KPdr
Cgt
KPgt
コピー
Cgt
Kgt
メーカー
Kdrm ライセンス
Kprd, 'PR'
DRM ライセンス
Kc, 'X'
(2)
(4)
DRM パッケージ
保護データ
保護コード
(3)
GT ライセンス
復号
復号鍵及び
アクセス権を設定
(5)
GT10 (CPU)
Kdrm 復号
(6)
TRB:
及び
TLB
Key ID
Key
Process
Rights
517
Kc
517
X
518
Kprd
517
PR

図41

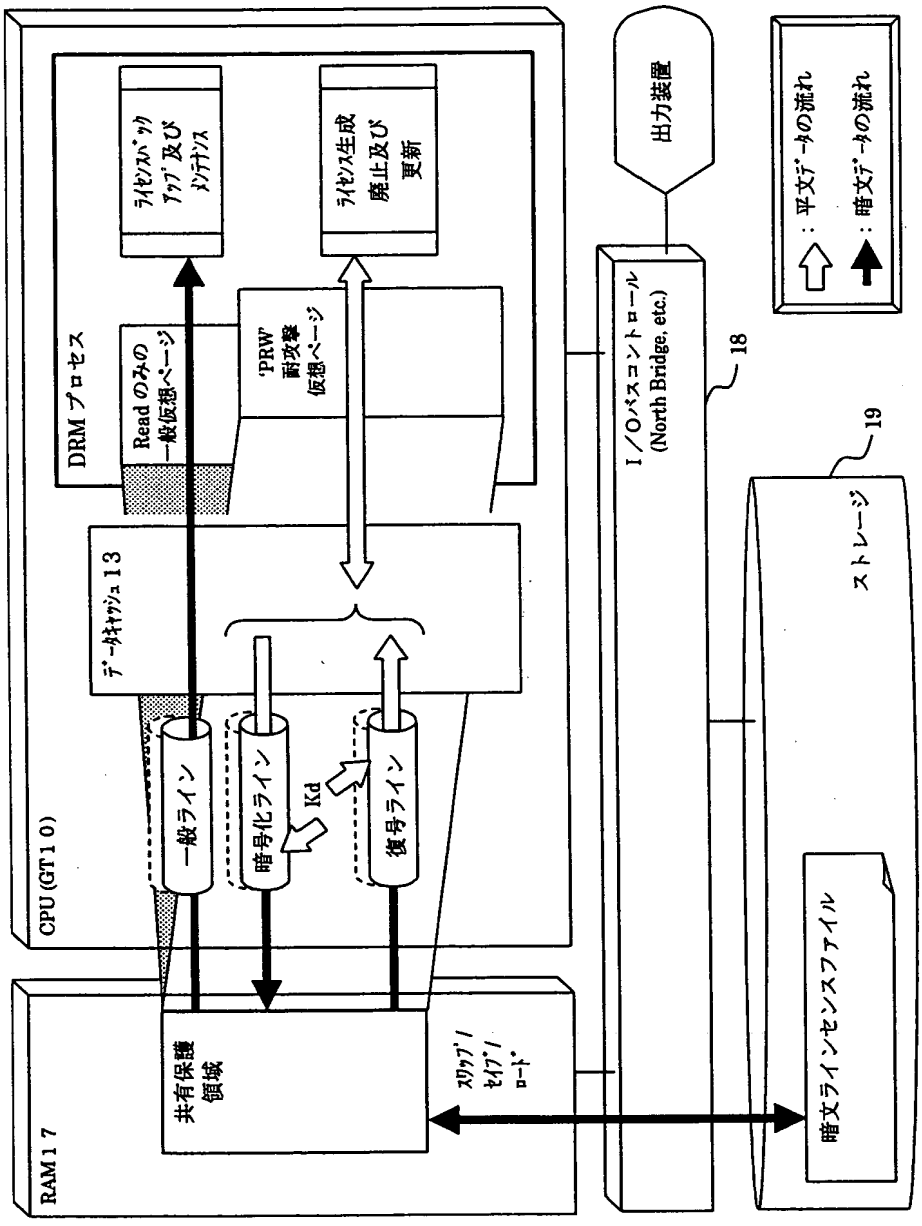
CPU (GT10)
DRM プロセス
ライセンスバックアップ及びメンテナンス
ライセンス生成廃止及び更新
Read のみの一般仮想ページ
'PRW' 耐攻撃仮想ページ
データキャッシュ13
一般ライン
暗号化ライン
Kd
復号ライン
RAM17
共有保護領域
スリップ/セイブ/ロード
I／Oバスコントロール (North Bridge, etc.)
18
出力装置
ストレージ
19
暗文ライセンスファイル
：平文データの流れ
：暗文データの流れ

図42

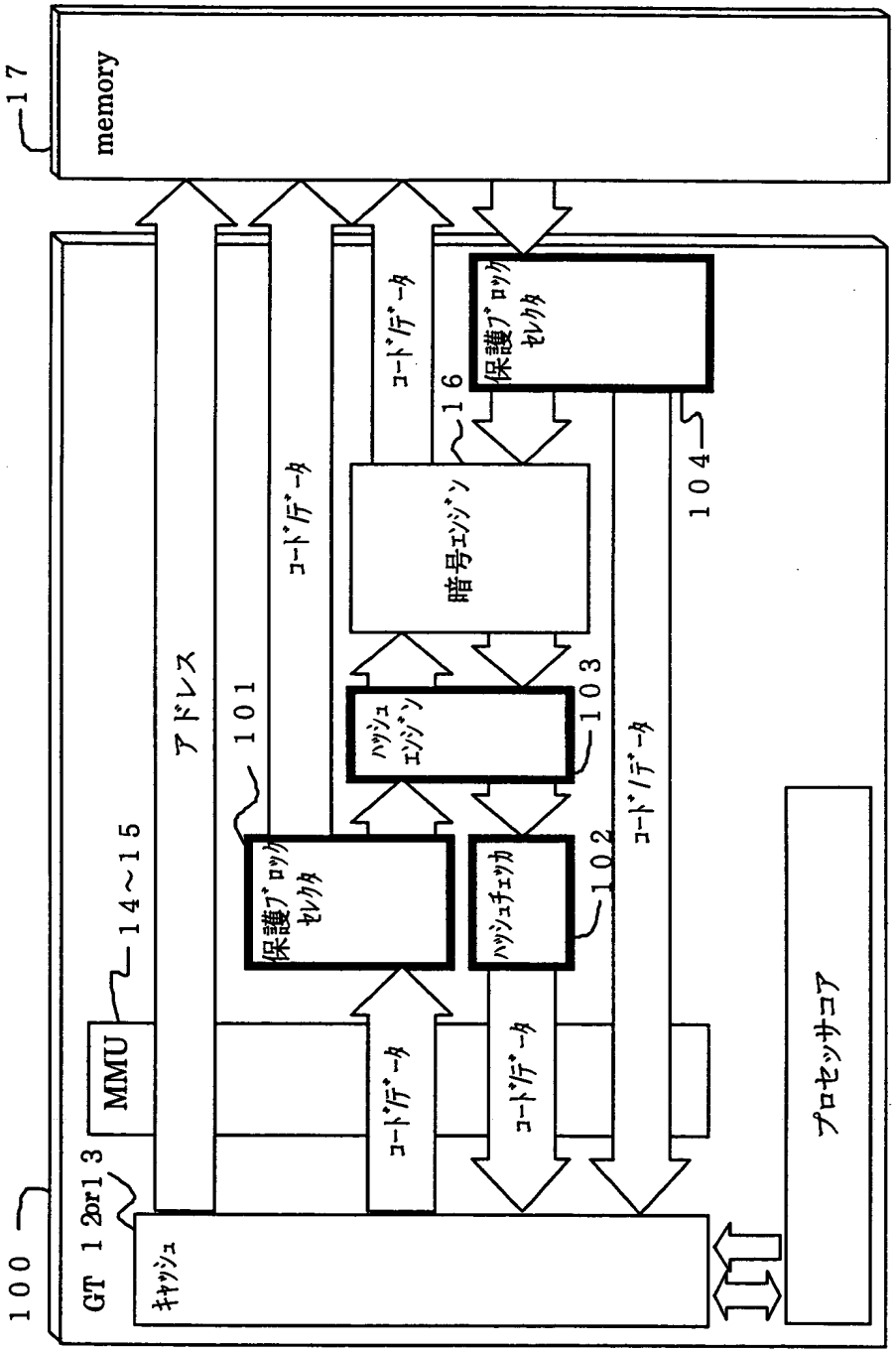
memory
17
100
GT 12or13
キャッシュ
MMU
14～15
アドレス
101
保護ブロック セレクタ
コードデータ
ハッシュ エンジン
暗号エンジン
16
保護ブロック セレクタ
104
103
ハッシュチェック
102
コードデータ
コードデータ
コードデータ
プロセッサコア

図43

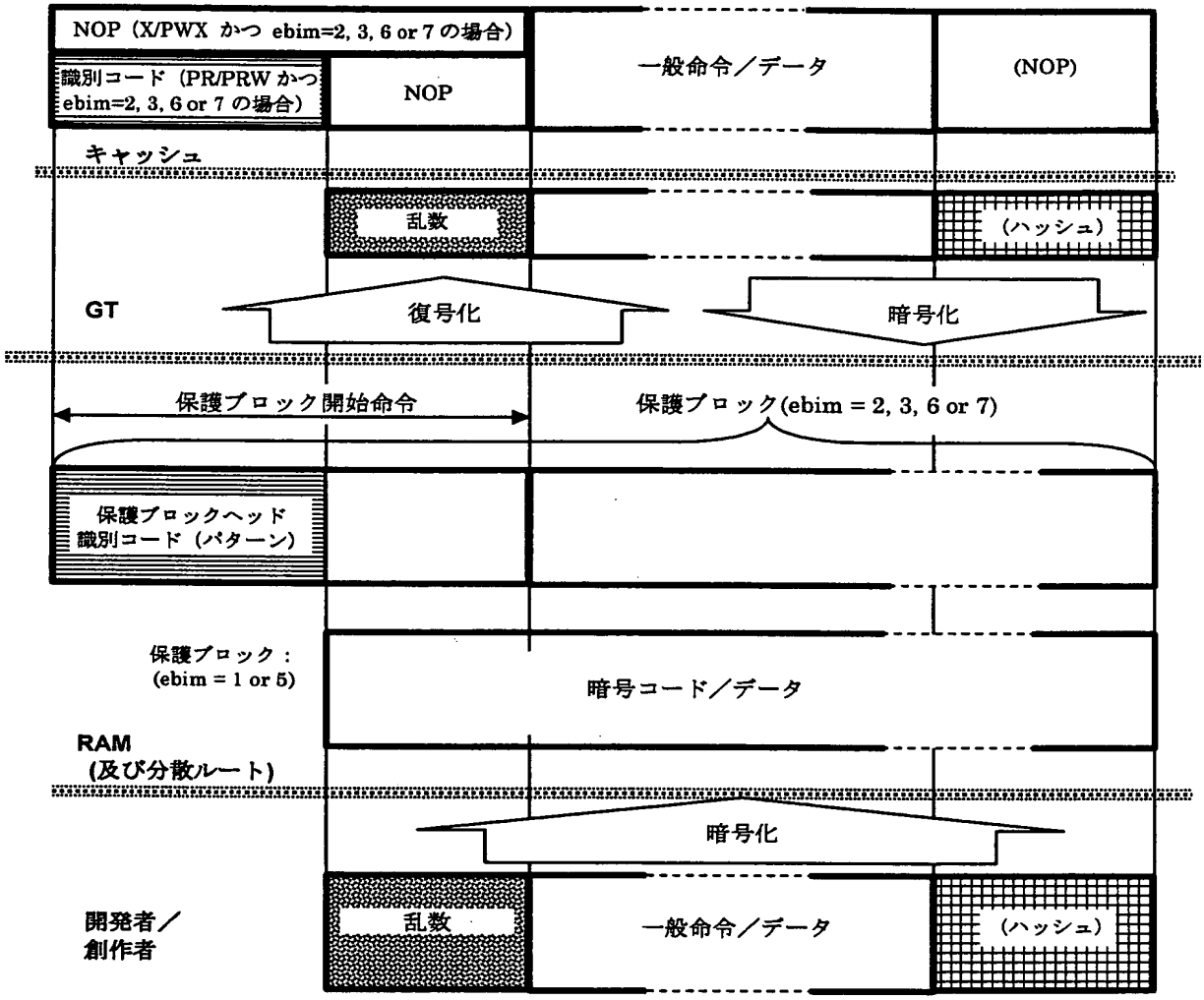
NOP（X/PWX かつ ebim=2, 3, 6 or 7 の場合）
識別コード（PR/PRW かつ ebim=2, 3, 6 or 7 の場合）
NOP
一般命令／データ
(NOP)
キャッシュ
乱数
(ハッシュ)
GT
復号化
暗号化
保護ブロック開始命令
保護ブロック(ebim = 2, 3, 6 or 7)
保護ブロックヘッド
識別コード（パターン）
保護ブロック：
(ebim = 1 or 5)
暗号コード／データ
RAM
(及び分散ルート)
暗号化
開発者／
創作者
乱数
一般命令／データ
(ハッシュ)

図４４

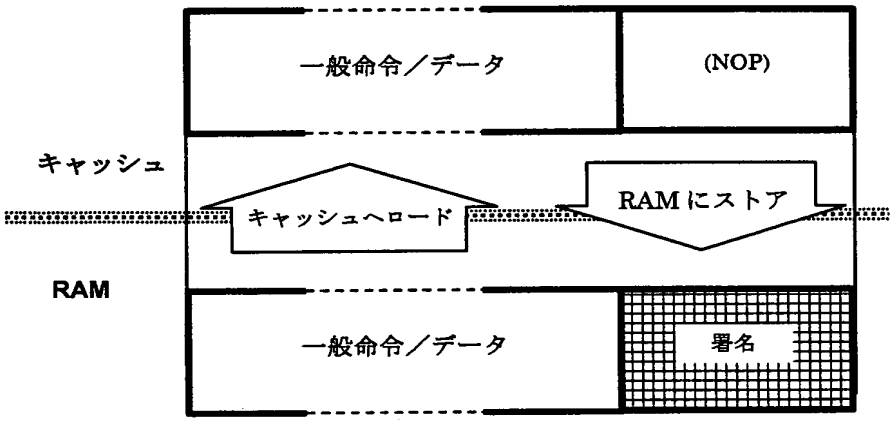

図４５

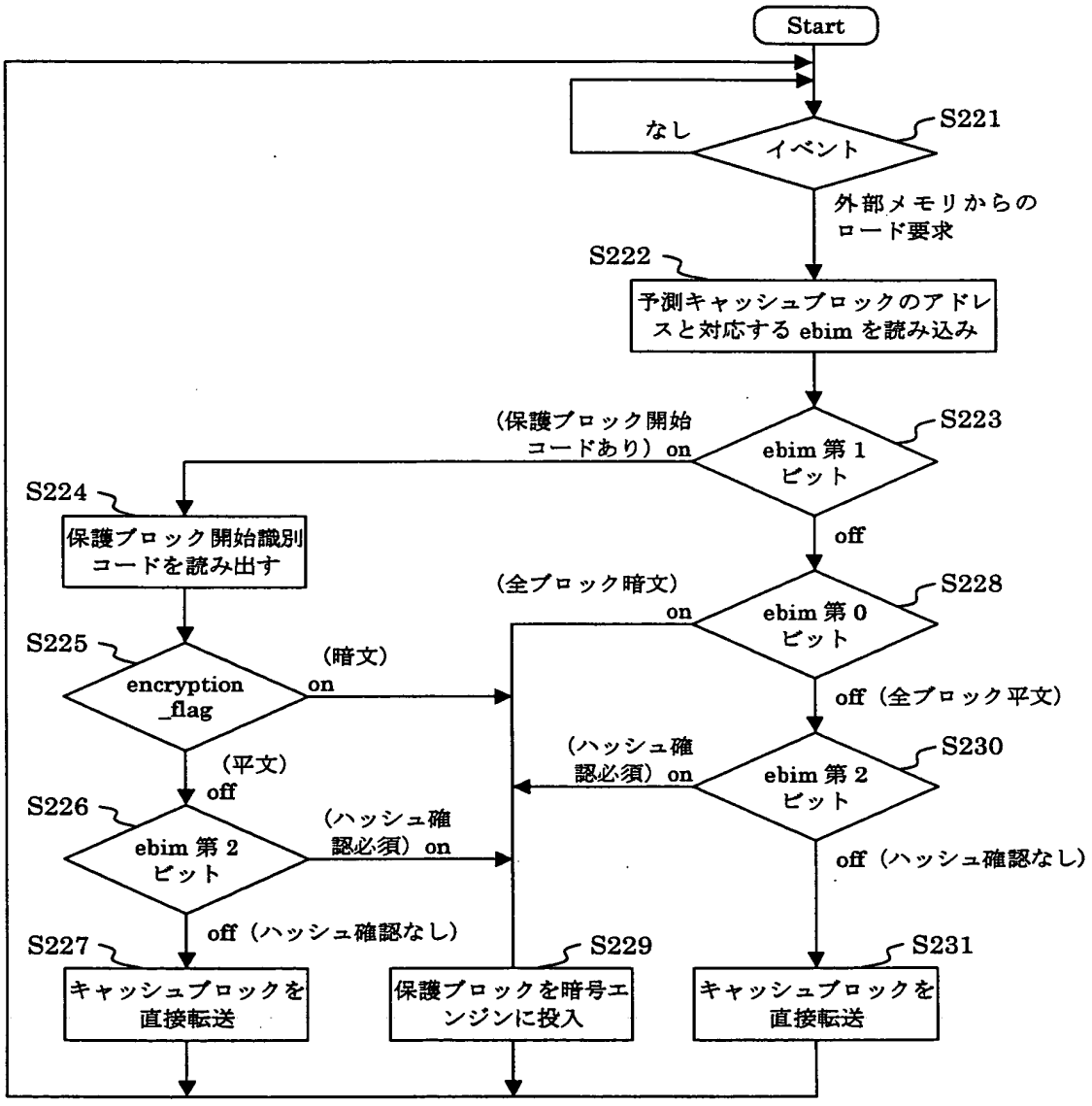
Start
なし
イベント
S221
外部メモリからの
ロード要求
S222
予測キャッシュブロックのアドレスと対応する ebim を読み込み
(保護ブロック開始
コードあり) on
ebim 第 1
ビット
S223
off
S224
保護ブロック開始識別
コードを読み出す
(全ブロック暗文)
on
ebim 第 0
ビット
S228
S225
encryption
_flag
(暗文)
on
off (全ブロック平文)
(平文)
off
(ハッシュ確
認必須) on
ebim 第 2
ビット
S230
S226
ebim 第 2
ビット
(ハッシュ確
認必須) on
off (ハッシュ確認なし)
off (ハッシュ確認なし)
S227
S229
S231
キャッシュブロックを
直接転送
保護ブロックを暗号エ
ンジンに投入
キャッシュブロックを
直接転送

図４６

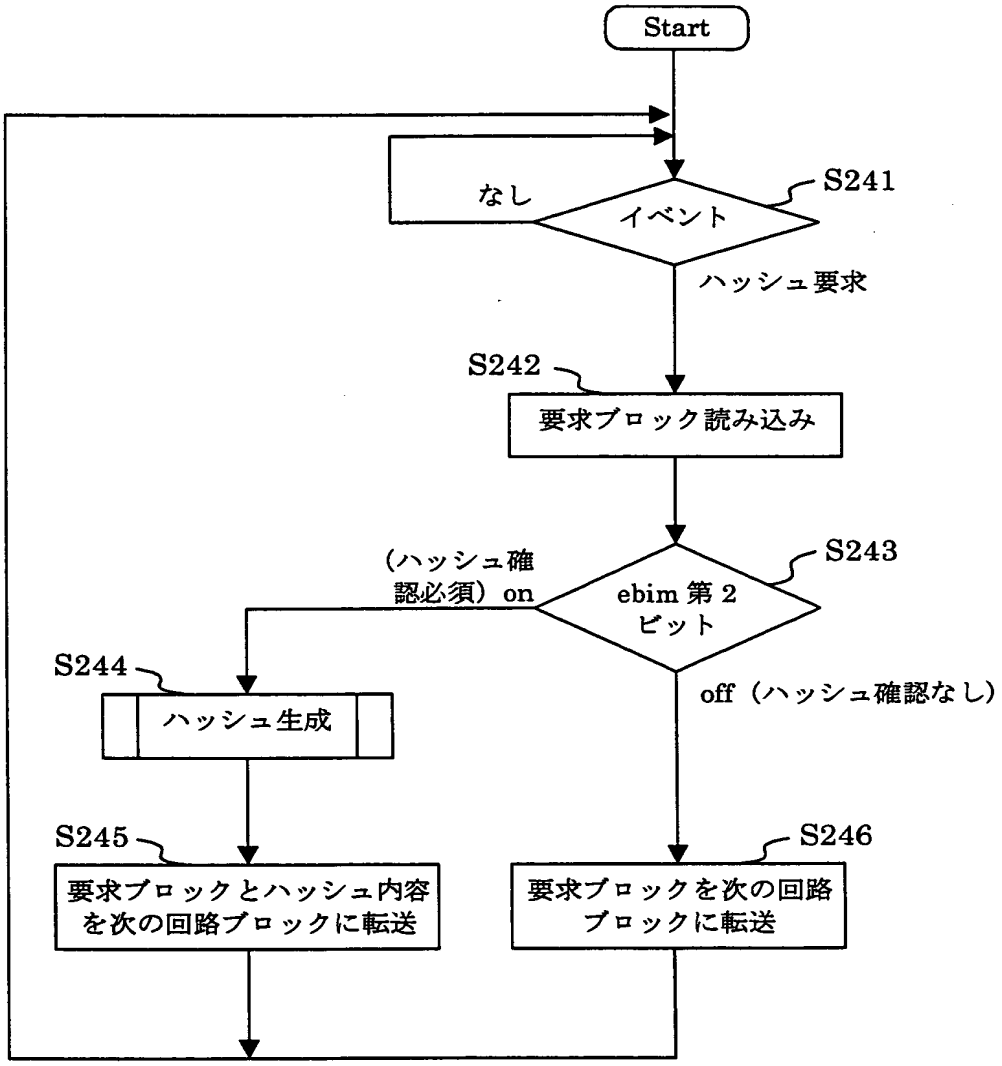
Start
なし
イベント
S241
ハッシュ要求
S242
要求ブロック読み込み
(ハッシュ確認必須) on
ebim 第 2 ビット
S243
S244
ハッシュ生成
off (ハッシュ確認なし)
S245
要求ブロックとハッシュ内容を次の回路ブロックに転送
S246
要求ブロックを次の回路ブロックに転送

図４７

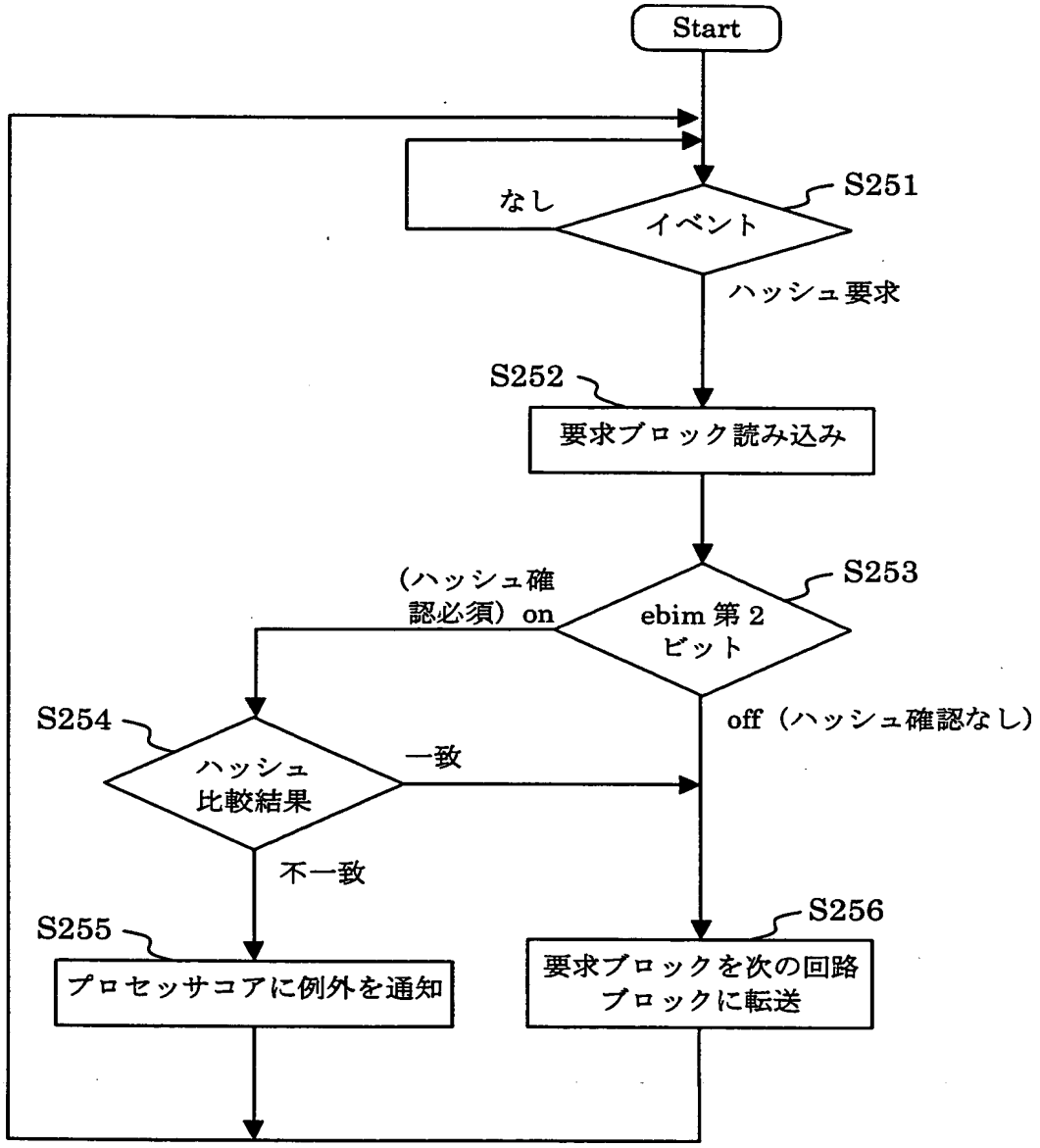
Start
なし
イベント
S251
ハッシュ要求
S252
要求ブロック読み込み
(ハッシュ確認必須) on
ebim 第 2 ビット
S253
off（ハッシュ確認なし）
S254
ハッシュ比較結果
一致
不一致
S255
プロセッサコアに例外を通知
S256
要求ブロックを次の回路ブロックに転送

図４８

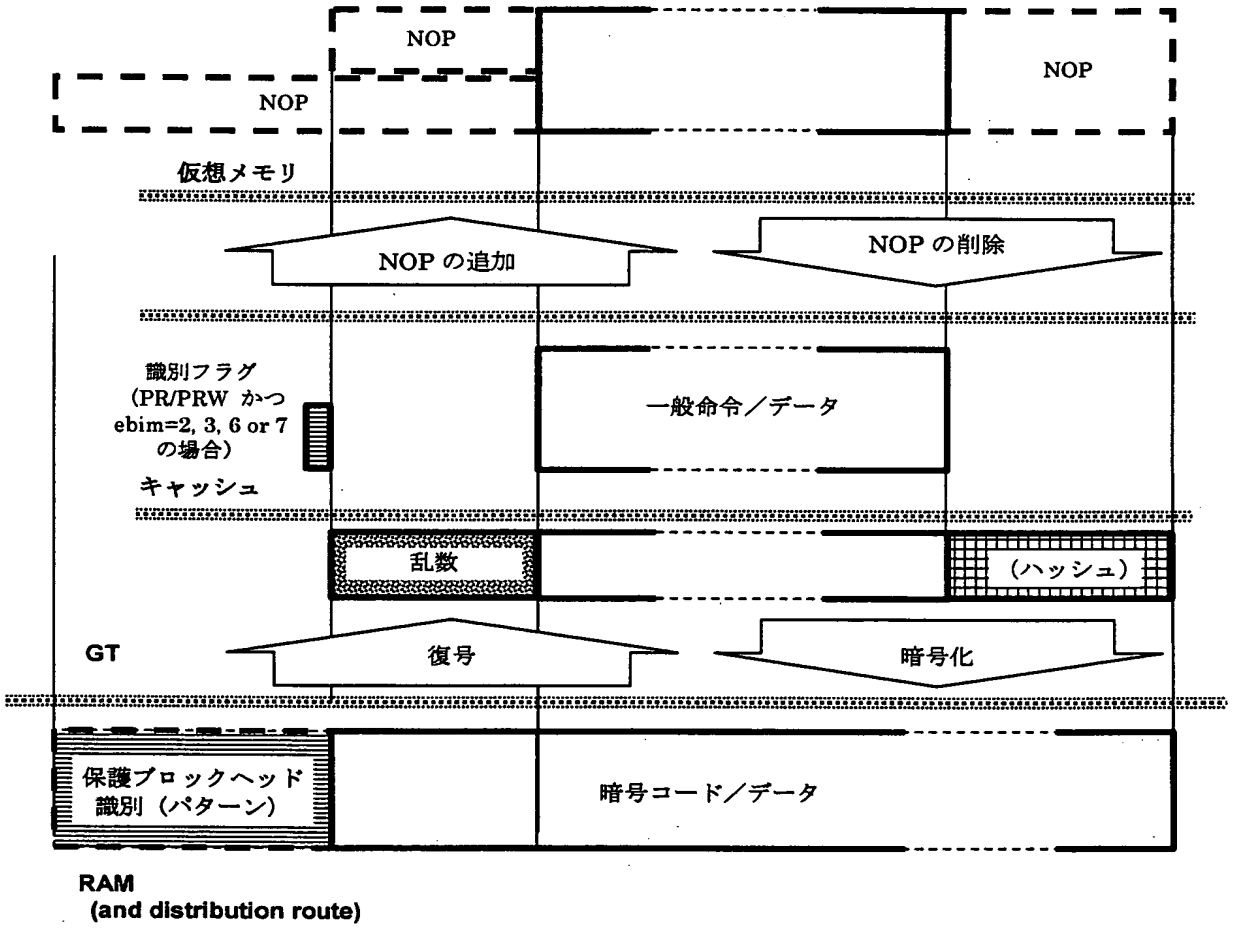
NOP
NOP
NOP
仮想メモリ
NOP の追加
NOP の削除
識別フラグ
(PR/PRW かつ
ebim=2, 3, 6 or 7
の場合)
一般命令／データ
キャッシュ
乱数
(ハッシュ)
GT
復号
暗号化
保護ブロックヘッド
識別（パターン）
暗号コード／データ
RAM
(and distribution route)

図４９

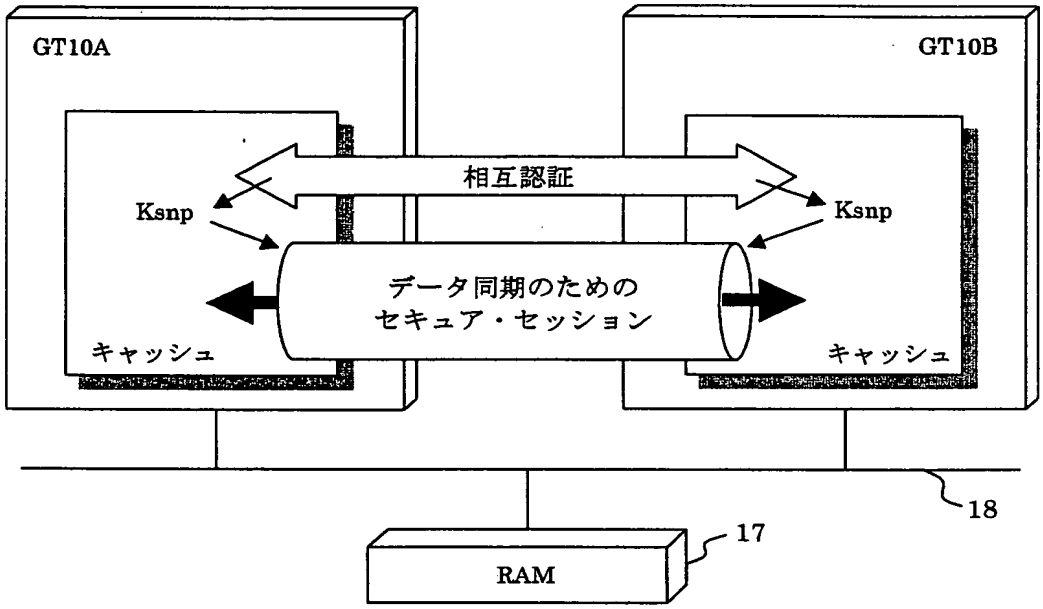
GT10A
GT10B
相互認証
Ksnp
Ksnp
データ同期のための
セキュア・セッション
キャッシュ
キャッシュ
18
17
RAM

図５０

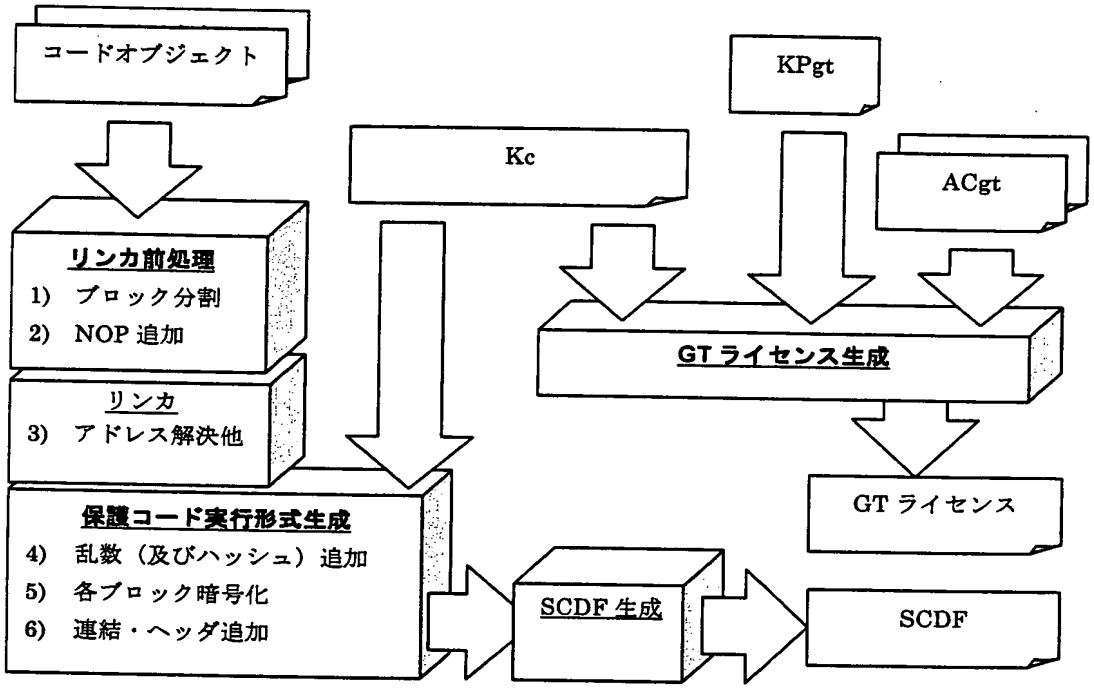
コードオブジェクト
KPgt
Kc
ACgt
リンカ前処理
1) ブロック分割
2) NOP 追加
GT ライセンス生成
リンカ
3) アドレス解決他
GT ライセンス
保護コード実行形式生成
4) 乱数（及びハッシュ）追加
5) 各ブロック暗号化
6) 連結・ヘッダ追加
SCDF 生成
SCDF

図５１

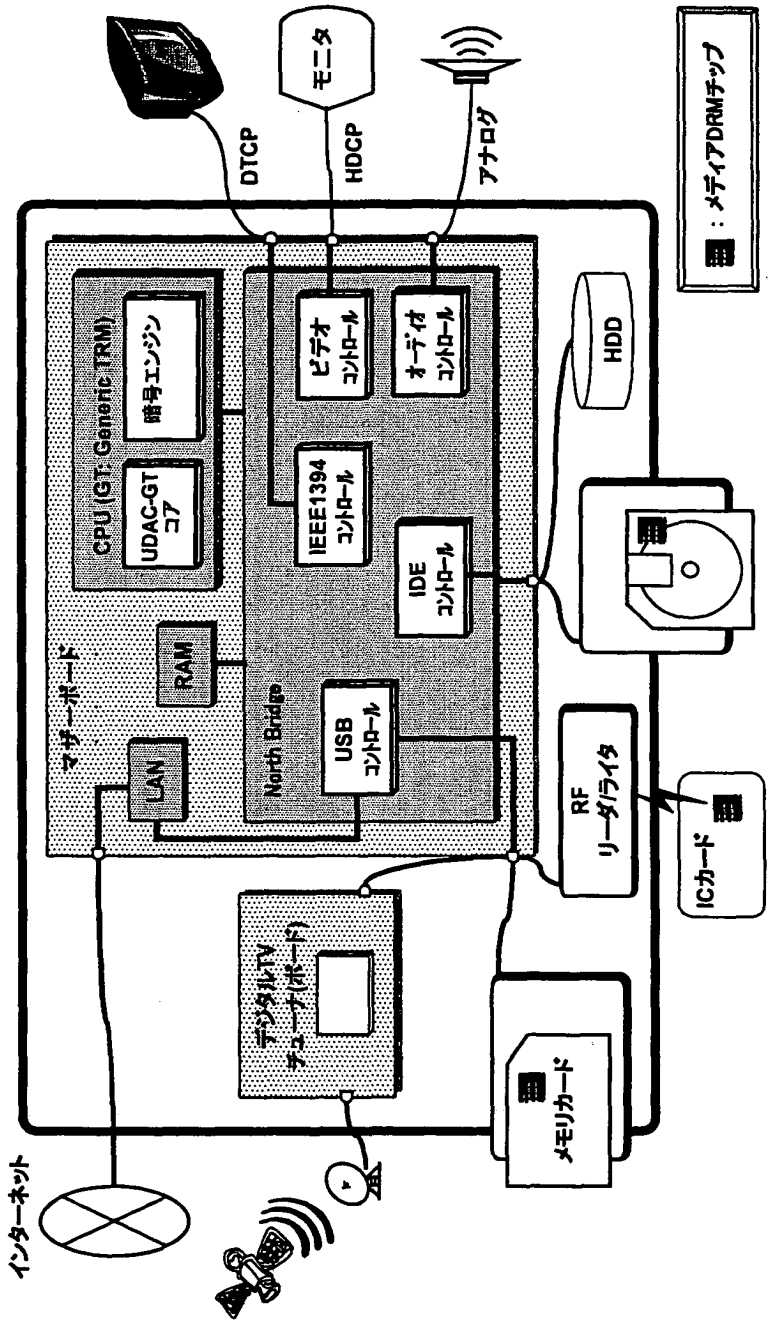
DTCP
HDCP
アナログ
モニタ
：メディアDRMチップ
CPU (GT: Generic TRM)
暗号エンジン
UDAC-GT コア
RAM
LAN
マザーボード
North Bridge
ビデオ コントロール
オーディオ コントロール
IEEE1394 コントロール
IDE コントロール
USB コントロール
HDD
RF リーダ/ライタ
ICカード
デジタルTV チューナ(ボード)
メモリカード
インターネット

図52